PEO Soldier Simulation Road Map VI

Command and Control Modeling

LTC Robert. H. Kewley, Jr., Ph.D., US Army
Director, Operations Research Center

Dr. Andreas Tolk, Ph.D.
Associate Professor, Engineering Management and Systems Engineering,
Old Dominion University

Sponsored by
Program Executive Office - Soldier

Operations Research Center of Excellence
West Point, NY 10996
November 2009

ORCEN Technical Report: DSE-TR-0906
DTIC: ADA509285

Executive Summary

Problem Definition The Army's Program Executive Office (PEO) - Soldier has the complex task of acquiring and integrating a system of soldier equipment that meets validated soldier mission requirements. In order to better assess trade-offs in different soldier architectures, they seek an improved simulation capability that better represents the individual soldier on the battlefield. No single model provides this capability. They are pursuing a strategy of integrating three different simulation models to take advantage of the strengths of each. These models are the Infantry Warrior Simulation (IWARS), One Semi-Automated Forces (OneSAF), and the Combined-Arms Analysis Tool for the 21st Century (COMBAT^{XXI}). After achieving real-time integration of the models using the High Level Architecture (HLA), this year's focus was to leverage that architecture in order to simulate information-enabled command and control for the dismounted fight. This modeling capability opens the door for analysis of the unit-level effects of ground soldier command and control equipment such as the Land Warrior or Ground Soldier System (GSS).

Technical Approach The approach to this modeling integration was to break down the overarching integration task into a series of discrete tasks that could be performed by model development teams involved in this project.

- Build automatic federation start/stop capabilities into the models.
- Enable IWARS dismounts to be carried on OneSAF or COMBAT ^{XXI} vehicles.
- Integrate a validated communications model into the architecture to assess communications effects given certain radio capabilities, environmental characteristics, and tactical distances between soldiers.
- Build fire support modeling into the federation.
- Build situation awareness modeling into the federation.
- Build command and control decision making into the federation.
- Model soldier casualties at a higher level of resolution in order to better assess effectiveness of different soldier helmets and body armor.
- Adapt the search and target acquisition process within the model to better capture soldier capabilities such as glimpsing or cuing.
- Model soldier power consumption during the mission to better capture the effects of different batteries or power consumption profiles of soldier equipment.
- Build identification-friend-or-foe (IFF) capabilities into the model to capture the effects of soldier-level IFF systems.

Results The focus of this year's efforts was to develop a federated simulation capability for dismounted command and control. This capability is built upon Research, Development, and Engineering Command's (RDECOM) Modeling Architecture for Technology, Research, and Experimentation (MATREX) and their Battle Command Management Services (BCMS). Federation management and design was supported by a systems engineering process for development of federated simulations. The specification borrows ideas from the systems engineering domain,

the simulation interoperability domain, and the software engineering domain. It has activities on the operational level to define how dismounted soldiers operate, the system level to define the overall architecture of the federation, and the technical level to fully specify requirement to model developers. Use of this process allowed developers to separate operational and systems concerns from their technical implementations and communicate more effectively in the design process. They coded their implementations more efficiently because the federation design and test cases communicated the requirements more completely. These requirements were fully traceable to both the operational capabilities represented and the analysis requirements.

With respect to federation of the models, integration was achieved with the following approaches:

Information Exchange System RDECOM's MATREX architecture was used to ensure interoperability in other RDECOM and Training and Doctrine Command (TRADOC) federations.

Environmental Representation In order to ensure the federation could run without having to generate multiple correlated terrain databases, each model was required to run natively using only OneSAF's Environmental Runtime Component (ERC).

Entity Representation Entity types were coordinated between models using equivalent weapons and Army Materiel Systems Analysis Activity (AMSAA) data. Appropriate MATREX object types were used to coordinate entity states during federation execution for dismounted soldiers, vehicles, and aircraft.

Models The models used common AMSAA validated target acquisition, direct fire, and indirect fire lethality models. Soldier casualties and movement were represented with a higher level of fidelity in IWARS.

Data Collection Data collection for analysis employed a mixed strategy. The internal data collection capabilities of each model were leveraged where possible, however some data required network based data collection tools that worked at the federation level.

Time Management The federation synchronized time by requiring each federate to run in real time.

By the end of the year, the federation demonstrated a significant capability to model dismounted command and control. BCMS tools maintained Situation awareness for all friendly dismounts and vehicles, regardless of which model actually owned the entity. This awareness was based upon the passing of situation reports and spot reports on the communications network - to include the calculation of communications effects. Based upon the situation awareness of small unit leaders, a fires federate automatically generated calls for fire against known enemy targets in the objective area. The system modeled the passing of fires messages and the clearance of fires process so that rounds were delivered only when the appropriate authority received the fires requests and cleared friendly forces from the target area using current situation awareness. In addition, a command and control federate used current situation awareness to synchronize the maneuver plan. Small unit leaders issued commands to execute subsequent phases and branches of the operations order by cross referencing their situation awareness against a decision support template for reactive maneuver decisions during the fight.

In addition to these command and control capabilities, the following advances were made:

- The Brigade and Below Propagation and Protocols (B2P2) model developed by Army Research Lab Survivability/Lethality Analysis Directorate (ARL-SLAD) began integration as a MATREX communications federate. Integration of this model is not yet complete. In addition, an integration effort has been started with the Communications Planner for Operational and Simulation Effects with Realism (COMPOSER) model developed by the Communications-Electronics Research, Development, and Engineering Center (CERDEC).

- Algorithms have been developed to capture dismounted casualty effects with higher resolution, especially with respect to fragmentation.
- Search and target acquisition algorithms have been developed to better represent soldier level glimpsing and cuing with an awareness of the operational environment.
- Power consumption algorithms have been developed for soldier equipment that allow for the disabling of equipment used by simulated soldiers when their batteries are dead.
- Initial studies have been conducted into the development and validation of IFF algorithms.

Given achievement of these capabilities, efforts for the upcoming year will focus on analysis tasks in support of the XM-25 Counter-Defilade Target Engagement System for Project Manager Individual Weapons and a terrain analysis capability for the Land Warrior and Ground Soldier System for Project Manager Ground Soldier. These analysis tasks will require representation of the XM-25 in each of the models. The modeling team will also develop decision models for route planning and position selection based on automated terrain and situation analysis. In addition, the federation will add America's Army as a game-based virtual federate that allows a human player to take on the role of a dismounted soldier in the scenario. As part of this integration, the ballistic dispersion and damage algorithms from OneSAF will be used to calculate ballistic and damage effects for the entities modeled in America's Army. This will aid verification and validation of this virtual federate.

Contents

1 Background

The PEO Soldier Simulation Road Map is an effort by PEO Soldier to develop within the Army a capability to model the effects of soldier equipment on unit-level effectiveness - focused at platoon and below. This study is the sixth year of collaborative effort between Program Executive Office (PEO) Soldier and the West Point Operations Research Center (ORCEN). Previous studies have led this effort to where it stands today. During the first year, the ORCEN analyzed simulation requirements and recommended a 3-model approach integrating IWARS, OneSAF, and COMBAT^{XXI}. During the second year, ORCEN effort was focused on establishing a memorandum of agreement between the three modeling agencies and mapping soldier equipment lists into prioritized modeling requirements. In the third year of effort, the modeling agencies signed the agreements and started prioritizing their work into common environmental and scenario representations that would enable "soft" linkages between the models. In the fourth year, these "soft" linkages were achieved, allowing scenarios to be run in one model, stopped, passed to a second model, and run to completion. In year five hard linkages were achieved allowing the models to exchange data during run-time. The sixth year leveraged these real-time linkages to achieve command and control modeling capabilities for the federation.

In November of 2003, Brigadier General James Moran, PEO Soldier, commissioned the ORCEN to develop a model, or family of models, that would support PEO Soldier decision making with respect to soldier equipment. The ORCEN, working within the PEO, further defined the need as, "PEO Soldier needs a simulation that allows the evaluation of platoon effectiveness based upon changes in Soldier tactical mission system (STMS) characteristics." Fulfillment of this need would bring the PEO in line with the Army's Simulation and Modeling for Acquisition, Requirements, and Training (SMART) program. The SMART program "involves rapid prototyping using M&S [modeling and simulation] media to facilitate systems engineering so that materiel systems meet users' needs in an affordable and timely manner while minimizing risk (AMSO, 2002)." Taking this need, the ORCEN evaluated a series of alternatives that ranged from creating a brand new simulation to adopting, in its entirety, an existing simulation. The team concluded that while developing a single model was cost and time prohibitive, no single existing model met the PEO's requirements. They recommended a federation of models including IWARS, OneSAF, and COMBAT^{XXI}. PEO Soldier accepted this recommendation and asked the ORCEN to lead the effort in building a team to develop this federation (Tollefson & Boylan, 2004).

While everyone understood the need for a federated modeling solution, the composition, type of integration, and level of detail for the federation were not so simple to agree upon. The ORCEN worked two parallel efforts from June 2004 until July 2005. First, they had to establish memoranda of agreement that would enable funding and collaboration within this project. This required significant negotiation between PEO Soldier, the Natick Soldier Center (developer of IWARS), PEO Simulation Training and Instrumentation (PEO-STRI - developer of OneSAF), and Training and Doctrine Command Analysis Center - White Sands Missile Range (TRAC-WSMR - developer of COMBAT^{XXI}). Second, they had to further refine the analysis requirements for the federation. In short, PEO Soldier did not have a list of analysis requirements; they had a list of equipment. The ORCEN worked with the PEO to categorize and streamline this list into a discrete set of modeling requirements that could be implemented by the members of the federation. Once these requirements were understood, it was easier for the modeling agencies to agree to develop these capabilities (Martin, 2005).

Given an agreement to work together, and a list of analysis needs, the next significant question is where to start. The modeling teams first came together under the signed agreements in 2005. However, there was not general agreement on the integration technology or on the initial analysis tasks. The ORCEN worked with the PEO to select from the list of analysis requirements, a very short list of equipment and associated analysis questions. Collectively, the group decided to begin effort on "soft" linkages. In other words, the models in the federation would not exchange data during run-time. Instead, they would agree on a common terrain representation and a common scenario representation. Using these representations, different models would take over the scenario, run a portion of the fight, update the status of the combatants, then pass that information to another model. Under this approach, the team could get started more quickly, develop a working relationship, and work out challenges to an eventual "hard" linkage where the models exchanged data with each other during the run. (Boylan, 2006).

During May of 2007, in the fourth year of effort for this project, the modeling team achieved a "soft" integration of two models, IWARS and COMBAT^{XXI}, for a small room-clearing scenario. This was made possible by the agreement between all of the development teams to use OneSAF's Environmental Runtime Component for common terrain and environment representation. They also agreed to use the Military Scenario Definition Language (MSDL) to share scenario data. Using this integration, the ORCEN analyst was able to collect mission performance data for the simulation run using a 2x2 factorial design. In this case, he represented two different levels of body armor and night vision equipment (Kramlich, 2007). This proof-of-concept integration was a major step in the five-year history of this project. The three models, selected in year 1, came together with a common understanding of the analysis requirements, established in year 2, and a common picture of the integration requirements, established in year 3. Most important in this successful linkage was the working relationships developed by the modeling teams and their commitment to the tasks at hand.

Much of the focus and effort for academic year 2008-2009 was centered on developing a working federation using High Level Architecture. A decision was made early in the year to use Research, Development, and Engineering Command's (RDECOM) Modeling Architecture for Technology, Research, and Experimentation (MATREX) for integration. Their federation architecture was designed to support the Future Combat Systems program, so it had the greatest support for advanced communications and command and control interactions. Both IWARS and OneSAF had already done development to support the MATREX federation object model. Another decision was made early in the year to adopt model driven architectures (MDA) to drive simulation development. In this manner, high-level activity diagrams represented the battlefield concepts. These were used to assign activities to different simulation models. More detailed sequence diagrams showed how the federation handled these activities using the technical details of the run-time infrastructure and federation object model. This communication enabled the modeling teams to better focus their efforts on code development. It also enabled explanation of these interactions to those who could not read the code. This aids verification and validation of the models, along with analysis. By the end of the year, the team had a working test scenario in which OneSAF controlled the vehicles and indirect fire elements, while IWARS controlled the dismounted forces moving into a village for a raid. This architecture supports analysis of the impact of soldier equipment, to include weapons, sensors, body armor, and communications gear, on the dismounted squad and its supporting mounted forces.

The successful HLA integration laid the groundwork for the 2008-2009 tasks for the Simulation Road Map. The task lists in ANNEX A highlight those efforts for each component model. In order to achieve those tasks, the West Point Department of Systems Engineering developed, in conjunction with the Virginia Modeling Analysis and Simulation Center (VMASC), two theoretical concepts to support federation development. The first of these is SysHub, a set of six engineering requirements for the development of federated simulation models in support of systems of systems analysis. A second supporting concept, a systems engineering process for the development of federated simulations, defined a process for meeting those engineering requirements. During the year, the development team followed this process. They began with an operational view that defined the scenario and the functional capabilities of the actual system. A systems view allowed allocation of functional tasks to supporting simulations within the federation. The technical view allowed further definition of the integration requirements via sequence diagrams that referenced the data structures within the MATREX federation object model. From these sequence diagrams, the ORCEN developed and distributed automated test cases to better define the integration requirements for each supporting simulation. The systems engineering process ensured traceability from the technical requirements all the way up to the operational functions and capabilities of the simulated system. By this process, the development team achieved federated simulation capabilities for situation awareness, communications, fires, and command and control. This allows analysis of the combat outcome that can be achieved with different dismounted command and control systems issued to different elements of the dismounted squad and platoon. This development was orchestrated in accordance with the project plan in ANNEX B.

2 SysHub - Engineering Requirements for Federated Simulation

Systems of systems integration is one of the biggest challenges facing military forces today. This challenge exists whether the integration occurs in the acquisition community or in the operational environment. Large acquisition programs such as PEO Soldier are doing parallel development of many systems. Architecture and design changes in one system have ripple effects that propagate across all of the systems in development and to the environment. This creates a complex network of interactions in which architectural changes to one of the subsystems impact other soldier systems, other systems into which soldiers must integrate, and soldiers who operate the system. A similar problem exists in the operational community. The evolving wartime environment demands a series of parallel training, acquisition, and fielding initiatives that must be integrated. Analysis of these interactions demands a more robust and reconfigurable simulation paradigm. The West Point Department of Systems Engineering's SysHub program looks to expand the research base and body of knowledge for modeling, simulation, and analysis to support systems of systems integration(Kewley *et al.* , 2008).

Federated simulations must be as agile, robust, and interactive as the operational environment they support. Large single models that enable analysis of systems of systems are not effective because no single modeling effort can account for all of the complexities. Instead, modeling and analysis must be done in a distributed and parallel fashion, mirroring the development of training, acquisition, and fielding initiatives. This leads to a number of subsystem models that effectively analyze different domains. In order to support systems of systems analysis, subsystem models need to be federated. Current federation architectures such as distributed interactive simulation (DIS) and existing versions of high level architecture (HLA) support interoperability in the military fire and engagement domain. Additional interoperability research is needed to support federations of models that examine other domains, such as command and control and human behavior. In addition, architectural modeling processes and tools that enable agile architecture development and revision will empower federation developers to ensure that interactions between models have shared meaning in both the conceptual and technical domains.

SysHub is a capability to rapidly conceptualize, develop, execute, and analyze data using federated simulation models. This concept is illustrated in Figure 1. At the core of this capability is a set of engineering capabilities. The outer ring shows a series of domains in which federated models have proven useful. Within that ring, we list a series of current application areas from those domains. For each application, a different federation of models must be developed to support the question at hand. These federates must be held together, conceptually and technically, by the engineering capabilities in the center of Figure 1. These are the key research areas that enable development of federated models:

Information Exchange System The capability to pass meaningful information between federates during the simulation run.

Environmental Representation The capability for federates to reference a shared and correlated environment in which entities interact.

Entity Representation The capability for federates to referenced shared conceptually aligned information about entities in the simulation. Some of this representation is passed via the information exchange system.

Models Within the context of the analysis or training question, the internal models of each federate must be validated and coordinated across the federation.

Data Collection The capability to collect meaningful information from the simulation run in the context of the analysis question or training objective for which the federation was designed.

Time Management The capability for all federates to maintain a common time reference in order to synchronize simulated events.

2.1 Information Exchange System

With respect to software systems, interoperability is "the ability of two or more systems or components to exchange information and to use the information that has been exchanged (Institute for Electrical and Electronics Engineers, 1990)." Data standards are specific agreements between agents responsible for different software subsystems that communicate with each other when functioning as parts of a larger system. An information exchange system provides for a basic level of interoperability between simulations. It is a necessary condition for any higher level of interoperability.

Two common models for providing interoperability are the Distributed Interactive Simulation (DIS) protocol (IEEE-SA Standards Board, 1998) and the High Level Architecture (HLA) (IEEE-SA Standards Board, 2000). These are established Institute for Electrical and Electronics Engineers (IEEE) standards for allowing simulations to share information at run time. An example of a data standard is the defined federation object model implemented to enable a specific federation instance over HLA. Unfortunately the techniques are generally brittle in one case (DIS) and challenging to implement in the other (HLA). In both cases the tools built to perform the underlying work, such as communication across the nework and parsing of messages, within a federation end up being in practice custom implementations built for a specific purpose.

The Army's experimentation program in support of transformation significantly challenged the existing federated simulation paradigm (Hurt *et al.* , 2006). This program had a need for the aggregation of engineering-level models in an environment that supported human-in-the-loop interaction and a robust command and control environment. The MATREX program took a systems engineering approach to attacking these challenges in a collective environment that spanned RDECOM in the context of systems of systems analysis and integration. They developed not only an HLA based information exchange system, but also the tools and procedures required to support integration and build community acceptance (Gallant *et al.* , 2009).

This project used used RDECOM's MATREX toolkit for integration. Their federation architecture was designed to support several other research and development programs, so it had the greatest support for advanced communications and command and control interactions. Both IWARS and OneSAF had already done development to support the MATREX federation object model. Support from the MATREX program has allowed this project to move from an idea to a reality in a relatively short fashion. Key MATREX aspects are:

Community acceptance Due to the fact that OneSAF and IWARS had already completed MATREX integration at some level for other exercises, it was a much easier sell to bring these programs into further integration using this architecture. In effect, MATREX has become a de facto standard for integration of Army simulations.

Cadre of simulation expertise Once we decided to perform MATREX integration, the MATREX support team was an invaluable source of expertise and training that allowed us to solve a variety of architecture and integration problems. They had already seen many of our problems in other communities and were quickly able to provide recommendations for solution.

Existing data model The existing data model, represented in the MATREX Federation Object Model, is a community accepted integration model that has met most of our integration needs with no modification.

Protocore middleware The Protocore middleware library has enabled our participating simulation de-

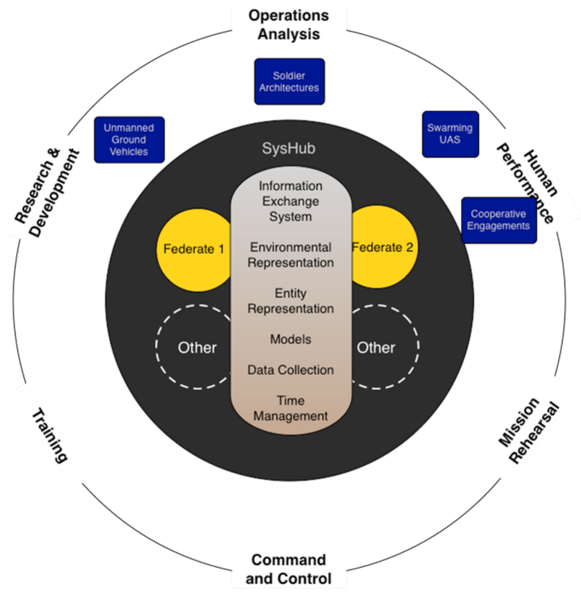


Figure 1: SysHub - Engineering capabilities for federated simulations in support of systems of systems analysis.

velopers to easily integrate their Java or C++ simulations without having to worry about the details of encoding and passing messages. As an anecdotal metric, the two simulations that use Protocore typically spend most of their time during our integration events dealing with the handling and interpretation of MATREX messages internal to their models. The two simulations that chose not to use Protocore are many months behind schedule and have still not successfully integrated at any of our events. They spend much of their time debugging the encoding.

Automated test cases The MATREX Automated Test Case tool allows us to distribute test cases with each new integration task. These test cases provide unambiguous messages and test capabilities to our developers which prevents costly misunderstandings.

Training support The MATREX team has offered training courses and on-site integration support that have allowed us to quickly learn and integrate their tools.

Battle command The MATREX Battle Command Management Services have allowed us to build a common operational picture that sits on the network as opposed to being internal to the simulations. This enables simulation-independent command and control and assessment of the knowledge state of entities in the exercise.

Government owned The lack of license fees and other charges makes MATREX affordable from both a cost and management perspective.

In summary, without MATREX, the integration would have taken up to six additional months, and the solution would be a fragile "point solution" based only on our internal knowledge and only on the needs of our project. We would have incurred additional costs by having to hire a commercial simulation integration company, and we would have adopted their tools – essentially locking us in to that commercial provider. The MATREX tools are a great solution for the information exchange system of any federation, but especially in the Army research and development domain.

2.2 Environmental Representation

In many simulation abstractions, entities interact with each other and with their environment. In federated simulations, it is not common for the models to share a single representation of the environment across the network. In most cases, each individual model has its own internal representation of that environment. If the results of the simulation are going to be valid, those environmental representations must be sufficiently correlated so that the results of the simulation are not obscured by differences in these representations.

Environment encompasses natural and man made physical elements of the battlespace occurring in the terrain, atmosphere, ocean, and space domains. In traditional modeling and simulation taxonomies, buildings and land-based infrastructure fall under the domain of terrain. This typically includes features that are considered more persistent over time. In addition, those things such as weather and obscurants that change over much smaller time scales are also important elements of environment that must be captured in its representation.

In federating different applications, it is key that the environment is correlated for meaningful interchange and execution of a scenario. This is a particularly challenging area based on the fact that there often are different means of defining the relief (elevation over spatial extents of the area of interest) and extracting and representing features. Different simulations often use different terrain data models for their internal representations.

Terrain databases created for different applications may or may not be created from the same source data. The source data may be of different resolutions and the techniques for sampling and integrating data may be different. Thus, it is not surprising that features, such as roads, bridges, rivers, and forested areas, that are the map for different applications do not nicely overlay each other. Additionally, if the surface relief is represented differently across applications, adjudication of line-of-sight and other interactions with the terrain become problematic. These can cause major issues when federating simulations. There is a need for entities in the simulation to have the same notion of where elements in the environment are and to execute interactions using a correlated representation of that environment.

The terrain data models include features such as road segment or surface element. These will have further attribution that denotes things such as type of road or soil type. Furthermore, there are allowable enumerations for road type and for soil type. Consider, soil type for example. Enumerations might be those denoted by the U.S. Geological Survey (USGS) soil classifications or they might be more simply represented as sand, clay, mud, or gravel. If linking applications using these two different soil type enumerations and considering ground vehicle mobility, it is important to understand how this affects the underlying algorithms computing mobility and how this affects a fair fight, interoperability, and correlation. There have been various efforts to standardize data. The Synthetic Environment Data Representation and Interchange Specification (SEDRIS) is an open project with several products and participates in standards development (SEDRIS, 2007).

The correlated generation of terrain information for simulation databases is a very broad and complex topic. There are three options with respect to terrain representation. The first option is to have each model interact with terrain using copies of the same terrain data model and associated programming interface for terrain algorithms such as mobility factors or line of sight. A second option is to develop correlated data from the same geospatial data sources in a single tool. There are several commercial and government-owned tools that use a common set of raw geospatial feature data, elevation data, and imagery to generate simulation databases in multiple formats. The final option is to independently generate correlated databases using different tools. This option may introduce correlation errors due to differences between the tools. It is possible to manually correlate the resulting databases using common reference points, but this is a difficult and imprecise process. Regardless of the option used, it is important to validate the effects of terrain data on the scenario during the federation test phase. Checking the validity of observed movement rates, line of sight algorithms, and visual representations will give confidence to the accuracy and correlation of environments across the federated models.

In order to simplify terrain representation and minimize the chances for negative impacts on the terrain algorithms, the PEO Soldier project has decided to require each federate to use OneSAF's Environmental Runtime Component (ERC) as its underlying data model. When a new scenario is required, there is no need to separately generate correlated copies of the terrain database in different formats. OneSAF, IWARS, and COMBAT XXI all natively reference the ERC when accessing terrain information during the run. The OneSAF team has supported this strategy by providing C++ wrappers for the Java-based ERC.

2.3 Entity Representation

In a discrete event simulation, entities interact with each other and their environment so that the system changes over time. A simulation's state as a "collection of variables necessary to describe a system at a particular time, relative to the objectives of a study (Law, 2007)." In this light, the state of an entity is the collective state of each of its attributes. Given the environment, interaction with other entities or changes in other entities that influence it, changed attributes represent a changed state. The challenge within the federation is timely and consistent application of each entity's state throughout.

In a federation, each supporting federate will likely use a different level of detail for the representation of each entity. Some models will require very detailed representations of their entities to support detailed calculations with respect to that entity's interaction with its environment and other entities. Other simulations may have a much more abstract representation with much less detail. The federation design must account for these differences. In addition, an entity may have a combination of static and dynamic attributes. The static attributes are set at simulation initialization and do not change over time. The type of weapon a particular soldier carries, for example, is often a static attribute. However, the dynamic attributes do change. In many cases, these changes must be communicated to other federates via the information exchange system. For example, a soldier's location and health status will change over the course of the fight, and these attributes will likely be very important to the entities represented in other federates.

The coordination takes place in three steps during federation design. Initially, a common reference description of each entity is identified. For example, a particular type of combat vehicle may be described by a fairly detailed document outlining its capabilities. The second step is for each federate to, as faithfully as possible, build a static and dynamic representation of that combat vehicle given the constraints of its internal data structures. Finally, for the dynamic attributes, each federate must define those attributes for which it requires notification when they are updated in another simulation. The federation designers must build a consolidated list of those attributes, and the information exchanges system must have data structures for representing changes to those attributes as simulation time advances.

Within the PEO Soldier federation, the important entities to be represented include dismounted soldiers, combat vehicles, and unmanned aircraft. The Military Scenario Definition Language (MSDL) is used to define the units and entities that will participate in the battle. During scenario development, the unit composition and entity capabilities are defined by authoritative TRADOC sources and by authoritative AMSAA weapons characteristics and data. These sources provide a common shared reference for the supporting combat models to build a scenario. IWARS represented dismounted soldiers in great detail, with very abstract representations of vehicles and aircraft. For OneSAF and IWARS, the aircraft and vehicles were represented in detail while the dismounted soldiers were represented in less detail than for IWARS. For the dynamic attributes, the MATREX objects IndividualCombatant, GroundPlatform, and AirPlatform were used to represent these entities (MATREX, 2008). Each federate subscribed to these interactions and updated the internal states of these entities as required.

2.4 Models

When composing models in a federation, model validation requires some particular considerations. A shallow application of interoperability might only seek to align models and federates by converting the simulation's data into the necessary input and output format specified by the federates. This can often be done at the programmer level. Support for interoperability protocols such as HLA and DIS often enable new federates to be "plugged in" with little or no programming or engineering. The danger in this type of integration is that the federated model may not be designed to perform analysis in the new context. Just because a model has been validated in one context does not mean that it can automatically be used properly in a new context. Composing valid federations requires some additional steps.

The subject matter experts for the federated model must work with the simulation systems engineers to validate the model in its new context. While model validation is a complex subject beyond the scope of this paper, a few key points aid the discussion. First, it is assumed that the federated model has already been validated in some context. If this is the case, the modeling assumptions from the original validation must be checked against the new simulation context to ensure that this new context does not invalidate these assumptions to a point where the model is not useful in the new context. Once this static validation is complete, further validation can take place during the federation test and evaluation phase. In this phase, data inputs and outputs from the model are checked against known quantities to ensure the model behaves properly in the new context. Finally, a subject matter expert in the domain to be modeled can visualize or check model runs within the new context to give "face validity" to the model in its new context.

An example of these considerations is the federation's handling of IWARS soldiers mounted inside OneSAF vehicles. Our implementation was for IWARS to mark the soldiers as mounted in the IndividualCombatant information exchange and move the locations of the soldiers to match the location of the combat vehicle. In each federate, entities were not permitted to directly engage those entities that were marked as mounted - they had to engage the vehicle. With respect to interoperability of the data structures, this approach worked perfectly well. The entities moved around the battlefield and dismounted at the proper location. However, the new context violated IWARS internal assumption that soldiers were always dismounted. The IWARS simulation, which owned the dismounts, had no internal model for assessing damage inside a vehicle. If a vehicle was hit, the federation was not valid because it would have not been possible to assess damage against its occupants. Fortunately, OneSAF did have an internal model for damage to occupants of vehicles. When a vehicle was hit, OneSAF ran this damage model against all of its occupants, regardless of which simulation owned those occupants. If one of the IWARS soldiers was damaged, OneSAF sent a DamageReport interaction to IWARS so that IWARS could update the damage state of its mounted entities. Another validation problem was that these mounted soldiers were still able to acquire enemy targets and build situation awareness, even though they were completely enclosed in a vehicle. IWARS had to turn off the target acquisition algorithm for these mounted soldiers because they could not see outside of the vehicles. This mount/dismount modeling challenge highlights the complexities that can arise when federating a simulation introduces a new context that may violate some of the modeling assumptions of the federates - in this case, IWARS' assumption that soldiers would always be dismounted.

2.5 Data Collection

Successful analysis requires a clear definition of the questions to be answered and an understanding of the extent to which the questions can be accurately represented and answered by the tools available to the analyst. There are two primary tasks involved in moving from questions to answers. First, the question must be translated to a form representable in the simulation, and criteria must be established for translating the results of the simulation back to answers to the original questions. Second, the simulation must be run and enough data collected to answer the question to the desired level of confidence.

In order to translate the questions as accurately as possible to a form answerable by simulation, the analyst must understand the modeling assumptions made by the designers of the simulation. A completely translated question must be in a form directly answerable by simulation. It should be phrased in terms of simulation inputs and outputs only, and a translation must be defined from its answers to answers to the original questions.

Once translated, the question establishes a requirement for data collection. If the tools available to the analyst do not support collection of the data required, tools must be adapted or developed to meet the need or the scope of the question must be reduced to fit the available data.

Current tools for data collection tend to fall into two broad categories. The first consists of standalone loggers which record raw data from DIS, HLA, or other data interchange systems. The other consists of integrated data collection and analysis systems that record information at a level of abstraction comparable to a simulation's internal object model. Integrated data collection systems are easier to use as long as they provide the data required. When other data is required, it is often possible to construct small ad-hoc tools to analyze and summarize the output of the lower-level loggers.

In addition to automated tools, human observation is a valuable tool for data collection. In some cases, the most practical way of capturing data may be to have a human expert watch or participate in events as they unfold. In others, human psychological or physiological performance may be an important part of the experiment, and must be measured. Thus visualization tools and immersive environments are also an important part of the analyst's data-collection toolset.

For the PEO Soldier project, three different approaches have been integrated in order to collect the required data for analysis. First, the internal data collection systems for IWARS, OneSAF, and Combat XXI are leveraged to collect data such as target acquisitions and weapons effects. A network based data collection tool collects HLA interactions and object updates in order to assess metrics such as soldier locations and unit attrition. Finally, a custom data collection federate was developed to monitor and record the situation awareness of soldiers on the battlefield. By this combination of tools, analysts were able to assess the necessary performance metrics and effectiveness metrics.

2.6 Time Management

In order to execute successfully, time must be synchronized across the federates so that the ordering of events and interactions mirrors the real system. Simulations are synchronous, so the federates must have a shared representation of time. There are several strategies to achieve this.

The first, and perhaps the simplest and most common, is to required the federates to execute in real time. In this case, each federate keeps an internal clock synchronized with real time and sends messages and state updates at the time they occur. The federations ordering of events is preserved with respect to real time. A modification of this approach is to have each federate run in a multiple of real time. This approach requires the least modification to the participating federates. However, the repeatability of the simulation cannot be guaranteed, and there is no way to know if one federate speeds up or slows down with respect to real time. There is a possibility for synchronization errors caused by a federate's ability to synchronize its internal clock with real time.

Another approach is to implement some form of time management at the federation level. There are several approaches to this, and they have different advantages and disadvantages (Taylor *et al.* , 2003)(Carothers *et al.* , 1997)(Fujimoto, 1998). These schemes all make trade offs with respect to performance, ease of implementation, and repeatability. The High Level Architecture implements a flexible approach that allows federates with different internal time advance mechanisms to participate in the same federation.

Because the MATREX Protocore libraries do not yet implement time management services at the federation level, the PEO Soldier project uses real time synchronization. Because of the complexity of IWARS, the performance penalty for real-time execution is fairly small. However, it is not possible for the current federation to guarantee repeatability in the time-ordering of its events.

3 Systems Engineering Process for the Development of Federated Simulations

The requirement for a systems engineering approach to federation development stems from the challenges of representing systems-of-systems integration in a simulation environment. Typically, simulations exist that represent the operation of some of the subsystems. However, when these subsystems are integrated to form a new capability, a new simulation model is often useful to support decision making and optimization with respect to that capability. The federation simulation developer must typically rely on a series of techniques from other disciplines and the expertise of supporting developers to design the federation. This paper outlines a systems engineering approach that may be used to guide the process (Kewley & Tolk, 2009).

The real challenge of simulation development is to ensure that the final product is consistent with the purposes for which the simulation project was originally started. It should support analysis of different strategies, represented as simulation inputs. In so doing, there must be a mechanism for tying different simulation development activities to the requirements for certain system functions to be modeled and to certain outputs to be produced. A systems engineering approach to simulation development ensures that these ties to requirements are maintained throughout the process.

The process borrows from the Model Driven Architectures (MDA) approach to produce models of the simulation system on three different levels —targeting three different sets of stakeholders (Object Management Group, 2007). Operational level activities produce an operational description of the system to be simulated. The operational products, analogous to the Computation Independent Model of MDA, are independent of the fact that a simulation model is being built. They consider the concerns of the operational stakeholders who must use or operate the system as it functions in the real world. The system level activities focus on the simulation architecture as a whole. These products, analogous to the Platform Independent Model of MDA, assign simulation functions and metrics to different simulation models. The primary stakeholder for system level activities are integration engineers and managers for each of the component models. The technical level activities focus on the detailed development of simulation components and the interfaces between them. These products, analogous to the Platform Specific Model of MDA, provide sufficient detail for software engineers to develop code that will interface with the overall simulation federation to provide the required results. In some cases, the software or test cases may be auto-generated from the technical specification.

3.1 Operational View Activities

In performing the operational level activities, simulation engineers are focused on the problem definition phase of the systems engineering process (Parnell *et al.*, 2008). Their primary goal during this phase is to gain an understanding, documented as a combination of Unified Modeling Language (UML) specifications and other diagrams, of the system to be modeled. They should understand its users, its functions, and the value derived from the operation of the system itself. The steps in Figure 2 represent the steps of this process.

Identify system use-cases Identify how the system is used by different classes of users to perform the function for which it was designed. This results in a high level UML use-case model for the system.

Functionally define the system Use the use-case model and work with system users to define in a hierarchy the functions of interest for the system. This results in a functional hierarchy for the system. Figure 3 shows this for the ground soldier command and control system.

Identify stakeholders Identify stakeholders for the system. These are not only users, but also system owners, system developers, and the client for

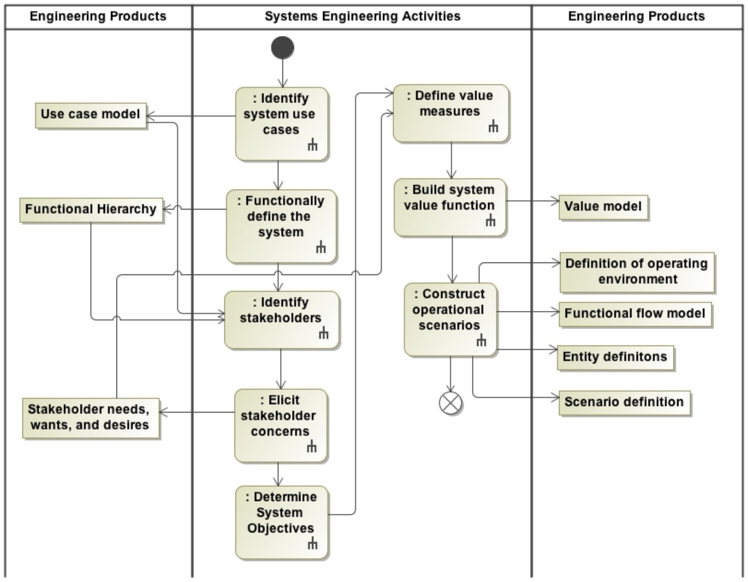

Engineering Products
Systems Engineering Activities
Engineering Products
Use case model
: Identify system use cases
: Define value measures
Functional Hierarchy
: Functionally define the system
: Build system value function
Value model
Definition of operating environment
: Identify stakeholders
: Construct operational scenarios
Functional flow model
Entity definitons
Scenario definition
Stakeholder needs, wants, and desires
: Elicit stakeholder concerns
: Determine System Objectives

Figure 2: Operational view activities

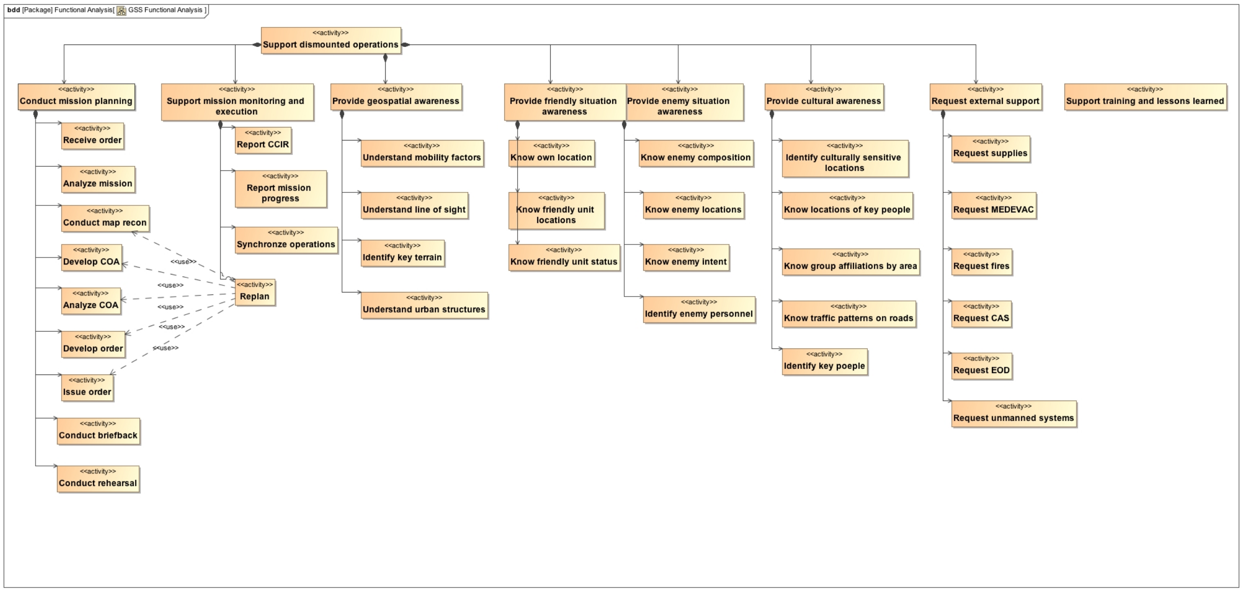


Figure 3: Functional hierarchy for ground soldier command and control system.

which the simulation study is being performed. Ensure the modeling detail captures enough information to answer the question at hand without incorporating so much detail that simulation development becomes overly difficult, expensive, and time consuming.

Elicit stakeholder concerns Using any combination of interviews, focus groups, and surveys, determine the fundamental concerns and values of the stakeholders identified in the previous steps.

Determine system objectives Based on the functional analysis and stakeholder concerns, determine the system objectives that must be successfully met in order to deliver value back to the stakeholders.

Define value measures Once the objectives have been identified, determine value measures that can be used to see if the system has indeed met those objectives. The simulation system, once developed, must be able to estimate system performance in terms of these objectives so that the relative performance of different alternatives can be determined. The results of this phase may be represented as a value hierarchy specified as a UML class diagram where each value measure is a component of an individual objective, and individual objectives are components of the overall system objective. It is also helpful to specify the range of possible values, from least desirable to most desirable, for each performance measure (Parnell *et al.* , 2008). Figure 4 shows the value hierarchy for a ground soldier command and control system.

Build system value function A simple value hierarchy is not sufficient for direct comparison between alternatives. The development team must return to the stakeholders and determine the value curves and weights for each performance measure, taking into consideration the importance and overall variability of each measure (Parnell *et al.* , 2008). This results in a value function that translates a set of performance scores for each alternative into an overall value score that represents the value of that alternative to the system stakeholders. Figure 5 shows the importance, variability, and associated swing weights of each of the value measures for the ground soldier command and control system.

Construct operational scenarios Once the system, its functions, and its values have been defined, the simulation study team must also define the scenario that represents the context for evaluation of system performance. In a military simulation, this represents details such as terrain and weather, forces in the engagement, supporting friendly forces, and full definitions of the entities in the simulation. The scenario definition describes the mission, forces involved, and roles of the simulated entities. In a military context, the Military Scenario Definition Language is an excellent standard for this representation (SISO, 2008). Figures 6 shows the MSDL representation of the platoon-level scenario used for the PEO Soldier project. This is a platoon-sized operation derived from the TRADOC Multi-Level Scenario 20 in which a Stryker platoon attacks to seize a series of buildings in a built up area. The primary focus of the operation is the first squad of the platoon that will secure Objective A. Entity definitions are an important aspect of scenario definition. All too often, the names of entities can lead to ambiguous understandings of the actual capabilities represented. Entity definitions should be as specific as possible with references to authoritative sources that provide accurate data to simulation modelers who must represent these entities in their respective simulations. Finally, within the scenario, the functions performed by the system under study should be defined as a functional flow model so that simulated events can be synchronized in the proper order. This model can be represented as a UML activity diagram. The functional flow model for the artillery fires request function of the ground soldier command and control system is shown in Figure 8.

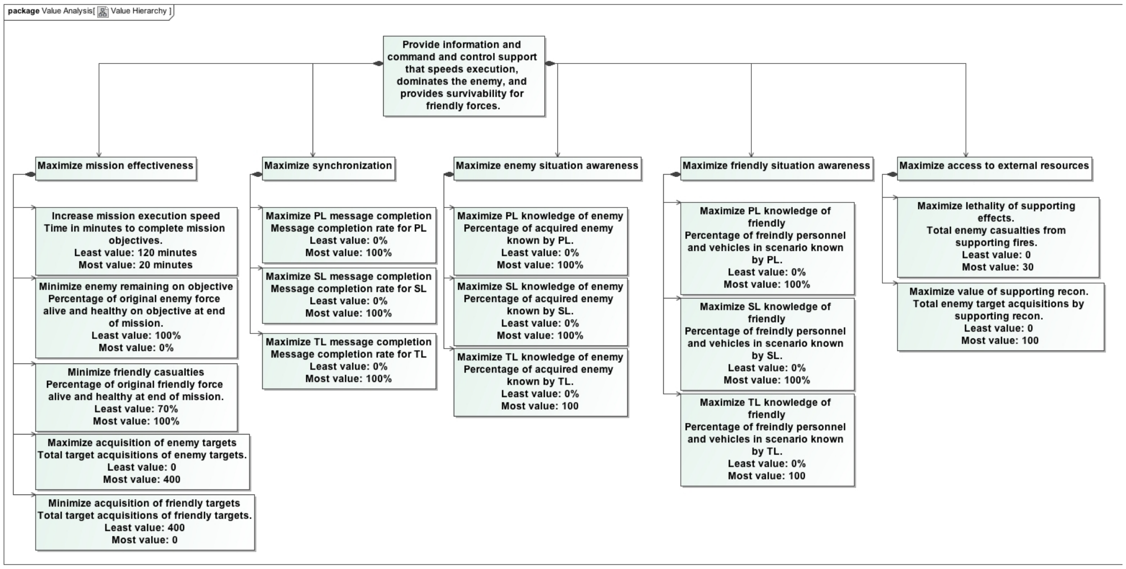
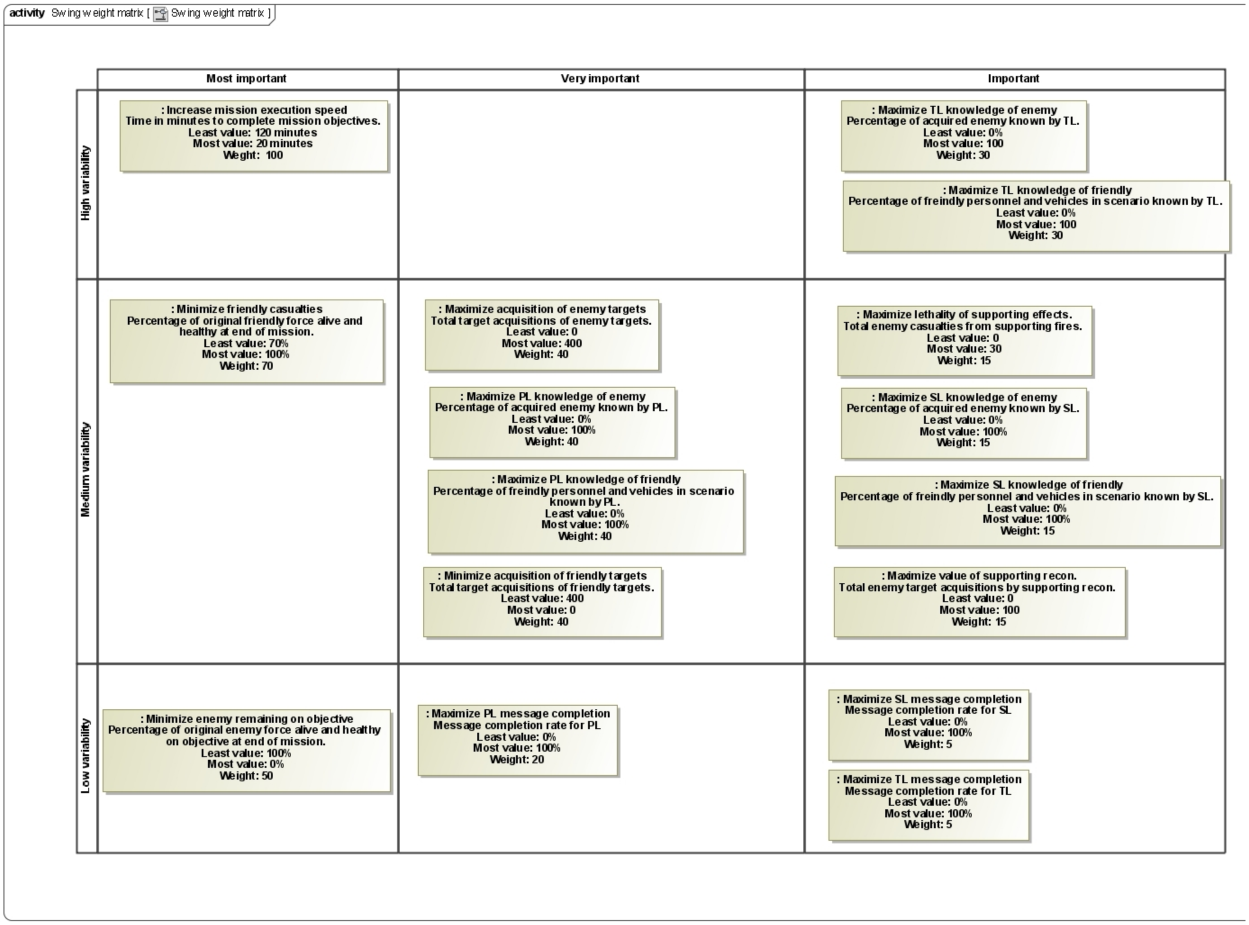

package Value Analysis[Value Hierarchy]
Provide information and command and control support that speeds execution, dominates the enemy, and provides survivability for friendly forces.
Maximize mission effectiveness
Increase mission execution speed
Time in minutes to complete mission objectives.
Least value: 120 minutes
Most value: 20 minutes
Minimize enemy remaining on objective
Percentage of original enemy force alive and healthy on objective at end of mission.
Least value: 100%
Most value: 0%
Minimize friendly casualties
Percentage of original friendly force alive and healthy at end of mission.
Least value: 70%
Most value: 100%
Maximize acquisition of enemy targets
Total target acquisitions of enemy targets.
Least value: 0
Most value: 400
Minimize acquisition of friendly targets
Total target acquisitions of friendly targets.
Least value: 400
Most value: 0
Maximize synchronization
Maximize PL message completion
Message completion rate for PL
Least value: 0%
Most value: 100%
Maximize SL message completion
Message completion rate for SL
Least value: 0%
Most value: 100%
Maximize TL message completion
Message completion rate for TL
Least value: 0%
Most value: 100%
Maximize enemy situation awareness
Maximize PL knowledge of enemy
Percentage of acquired enemy known by PL.
Least value: 0%
Most value: 100%
Maximize SL knowledge of enemy
Percentage of acquired enemy known by SL.
Least value: 0%
Most value: 100%
Maximize TL knowledge of enemy
Percentage of acquired enemy known by TL.
Least value: 0%
Most value: 100
Maximize friendly situation awareness
Maximize PL knowledge of friendly
Percentage of freindly personnel and vehicles in scenario known by PL.
Least value: 0%
Most value: 100%
Maximize SL knowledge of friendly
Percentage of freindly personnel and vehicles in scenario known by SL.
Least value: 0%
Most value: 100%
Maximize TL knowledge of friendly
Percentage of freindly personnel and vehicles in scenario known by TL.
Least value: 0%
Most value: 100
Maximize access to external resources
Maximize lethality of supporting effects.
Total enemy casualties from supporting fires.
Least value: 0
Most value: 30
Maximize value of supporting recon.
Total enemy target acquisitions by supporting recon.
Least value: 0
Most value: 100

Figure 4: Ground soldier command and control system value hierarchy.



activity Swing weight matrix [Swing weight matrix]

	Most important	Very important	Important
High variability	: Increase mission execution speed Time in minutes to complete mission objectives. Least value: 120 minutes Most value: 20 minutes Weght: 100		: Maximize TL knowledge of enemy Percentage of acquired enemy known by TL. Least value: 0% Most value: 100 Weight: 30 : Maximize TL knowledge of friendly Percentage of freindly personnel and vehicles in scenario known by TL. Least value: 0% Most value: 100 Weight: 30
Medium variability	: Minimize friendly casualties Percentage of original friendly force alive and healthy at end of mission. Least value: 70% Most value: 100% Weight: 70	: Maximize acquisition of enemy targets Total target acquisitions of enemy targets. Least value: 0 Most value: 400 Weight: 40 : Maximize PL knowledge of enemy Percentage of acquired enemy known by PL. Least value: 0% Most value: 100% Weight: 40 : Maximize PL knowledge of friendly Percentage of freindly personnel and vehicles in scenario known by PL. Least value: 0% Most value: 100% Weight: 40 : Minimize acquisition of friendly targets Total target acquisitions of friendly targets. Least value: 400 Most value: 0 Weight: 40	: Maximize lethality of supporting effects. Total enemy casualties from supporting fires. Least value: 0 Most value: 30 Weight: 15 : Maximize SL knowledge of enemy Percentage of acquired enemy known by SL. Least value: 0% Most value: 100% Weight: 15 : Maximize SL knowledge of friendly Percentage of freindly personnel and vehicles in scenario known by SL. Least value: 0% Most value: 100% Weight: 15 : Maximize value of supporting recon. Total enemy target acquisitions by supporting recon. Least value: 0 Most value: 100 Weight: 15
Low variability	: Minimize enemy remaining on objective Percentage of original enemy force alive and healthy on objective at end of mission. Least value: 100% Most value: 0% Weight: 50	: Maximize PL message completion Message completion rate for PL Least value: 0% Most value: 100% Weight: 20	: Maximize SL message completion Message completion rate for SL Least value: 0% Most value: 100% Weight: 5 : Maximize TL message completion Message completion rate for TL Least value: 0% Most value: 100% Weight: 5

Figure 5: Swing weight matrix used to build system value function.

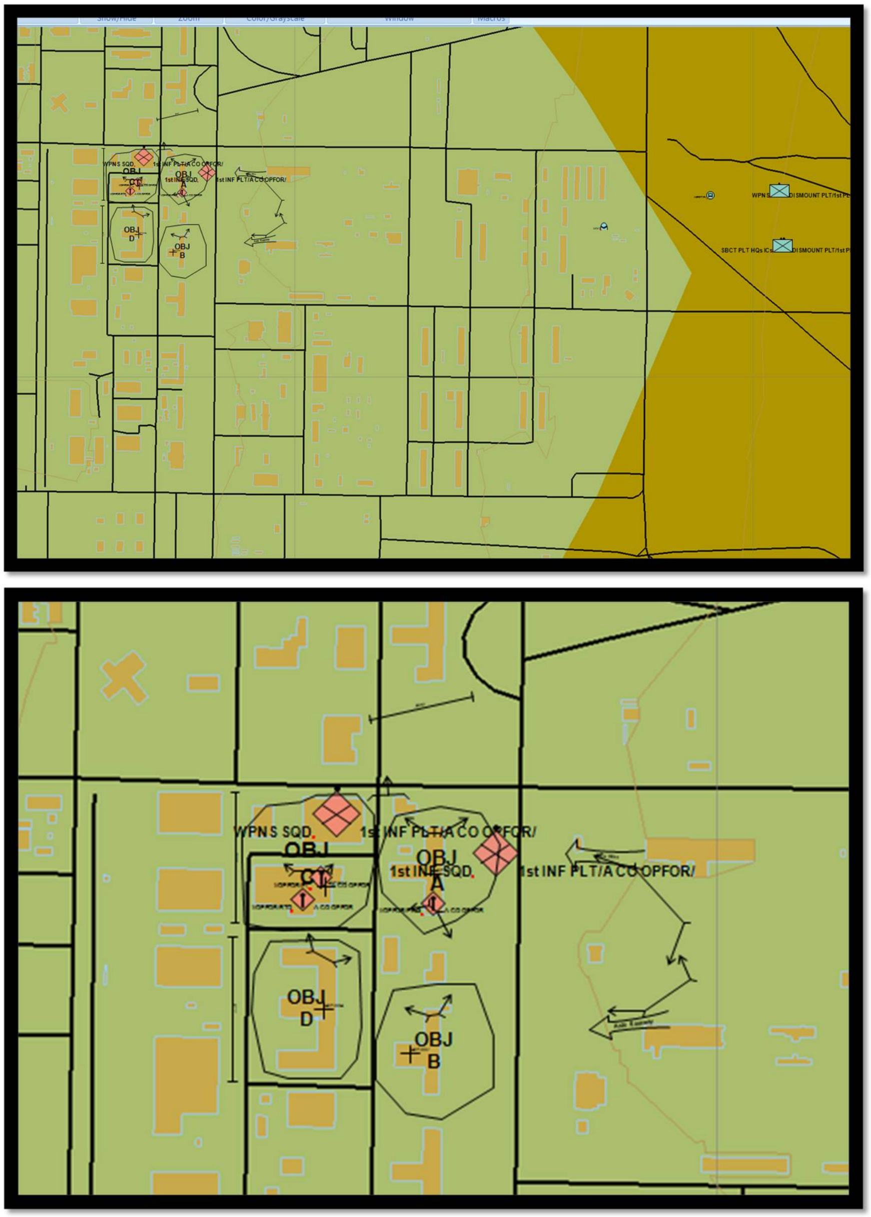
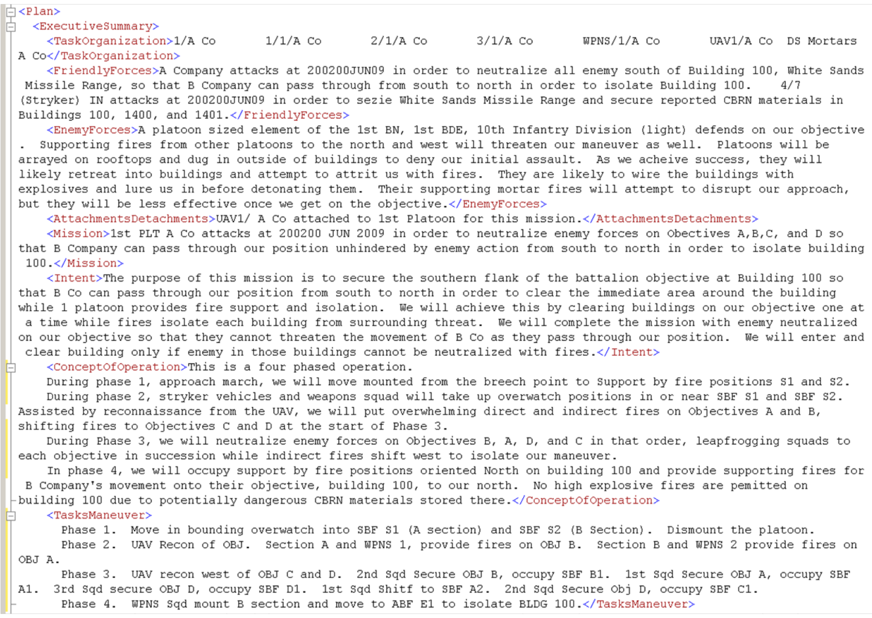


Figure 6: Military Scenario Definition Language representation of attack scenario.



```
<Plan>
  <ExecutiveSummary>
    <TaskOrganization>1/A Co       1/1/A Co       2/1/A Co       3/1/A Co       WPNS/1/A Co       UAV1/A Co  DS Mortars
A Co</TaskOrganization>
    <FriendlyForces>A Company attacks at 200200JUN09 in order to neutralize all enemy south of Building 100, White Sands
 Missile Range, so that B Company can pass through from south to north in order to isolate Building 100.    4/7
(Stryker) IN attacks at 200200JUN09 in order to sezie White Sands Missile Range and secure reported CBRN materials in
Buildings 100, 1400, and 1401.</FriendlyForces>
    <EnemyForces>A platoon sized element of the 1st BN, 1st BDE, 10th Infantry Division (light) defends on our objective
.  Supporting fires from other platoons to the north and west will threaten our maneuver as well.  Platoons will be
arrayed on rooftops and dug in outside of buildings to deny our initial assault.  As we acheive success, they will
likely retreat into buildings and attempt to attrit us with fires.  They are likely to wire the buildings with
explosives and lure us in before detonating them.  Their supporting mortar fires will attempt to disrupt our approach,
but they will be less effective once we get on the objective.</EnemyForces>
    <AttachmentsDetachments>UAV1/ A Co attached to 1st Platoon for this mission.</AttachmentsDetachments>
    <Mission>1st PLT A Co attacks at 200200 JUN 2009 in order to neutralize enemy forces on Obectives A,B,C, and D so
that B Company can pass through our position unhindered by enemy action from south to north in order to isolate building
 100.</Mission>
    <Intent>The purpose of this mission is to secure the southern flank of the battalion objective at Building 100 so
that B Co can pass through our position from south to north in order to clear the immediate area around the building
while 1 platoon provides fire support and isolation.  We will achieve this by clearing buildings on our objective one at
 a time while fires isolate each building from surrounding threat.  We will complete the mission with enemy neutralized
on our objective so that they cannot threaten the movement of B Co as they pass through our position.  We will enter and
 clear building only if enemy in those buildings cannot be neutralized with fires.</Intent>
    <ConceptOfOperation>This is a four phased operation.
    During phase 1, approach march, we will move mounted from the breech point to Support by fire positions S1 and S2.
    During phase 2, stryker vehicles and weapons squad will take up overwatch positions in or near SBF S1 and SBF S2.
Assisted by reconnaissance from the UAV, we will put overwhelming direct and indirect fires on Objectives A and B,
shifting fires to Objectives C and D at the start of Phase 3.
    During Phase 3, we will neutralize enemy forces on Objectives B, A, D, and C in that order, leapfrogging squads to
each objective in succession while indirect fires shift west to isolate our maneuver.
    In phase 4, we will occupy support by fire positions oriented North on building 100 and provide supporting fires for
 B Company's movement onto their objective, building 100, to our north.  No high explosive fires are pemitted on
building 100 due to potentially dangerous CBRN materials stored there.</ConceptOfOperation>
    <TasksManeuver>
      Phase 1.  Move in bounding overwatch into SBF S1 (A section) and SBF S2 (B Section).  Dismount the platoon.
      Phase 2.  UAV Recon of OBJ.  Section A and WPNS 1, provide fires on OBJ B.  Section B and WPNS 2 provide fires on
OBJ A.
      Phase 3.  UAV recon west of OBJ C and D.  2nd Sqd Secure OBJ B, occupy SBF B1.  1st Sqd Secure OBJ A, occupy SBF
A1.  3rd Sqd secure OBJ D, occupy SBF D1.  1st Sqd Shitf to SBF A2.  2nd Sqd Secure Obj D, occupy SBF C1.
      Phase 4.  WPNS Sqd mount B section and move to ABF E1 to isolate BLDG 100.</TasksManeuver>
```

Figure 7: XML representation of the operations order in military scenario definition language.

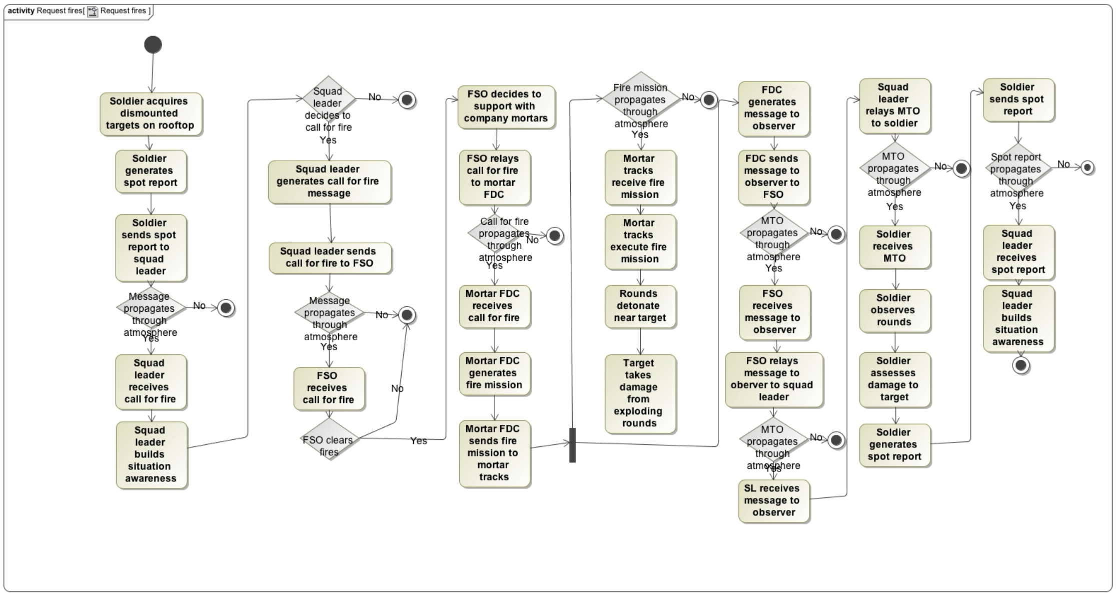

activity Request fires[Request fires]
Soldier acquires dismounted targets on rooftop
Soldier generates spot report
Soldier sends spot report to squad leader
Message propagates through atmosphere
No
Yes
Squad leader receives call for fire
Squad leader builds situation awareness
Squad leader decides to call for fire
No
Yes
Squad leader generates call for fire message
Squad leader sends call for fire to FSO
Message propagates through atmosphere
No
Yes
FSO receives call for fire
FSO clears fires
No
Yes
FSO decides to support with company mortars
FSO relays call for fire to mortar FDC
Call for fire propagates through atmosphere
No
Yes
Mortar FDC receives call for fire
Mortar FDC generates fire mission
Mortar FDC sends fire mission to mortar tracks
Fire mission propagates through atmosphere
No
Yes
Mortar tracks receive fire mission
Mortar tracks execute fire mission
Rounds detonate near target
Target takes damage from exploding rounds
FDC generates message to observer
FDC sends message to observer to FSO
MTO propagates through atmosphere
No
Yes
FSO receives message to observer
FSO relays message to oberver to squad leader
MTO propagates through atmosphere
No
Yes
SL receives message to observer
Squad leader relays MTO to soldier
MTO propagates through atmosphere
No
Yes
Soldier receives MTO
Soldier observes rounds
Soldier assesses damage to target
Soldier generates spot report
Soldier sends spot report
Spot report propagates through atmosphere
No
Yes
Squad leader receives spot report
Squad leader builds situation awareness

Figure 8: Call for fire functional flow model

3.2 Systems View Activities

Once the operational picture is well understood and documented, it is time for the high-level systems design of the federated simulation. The design steps in this phase build UML specifications of the simulation functions, simulation data collection, information exchanges, and environmental data. These steps result in a logical allocation of functionality and information exchanges that derive from the operational requirements from the previous step. While not sufficient for writing code, these models allow simulation engineers and software engineers from the participating federates to allocate high-level tasks and work packages to support simulation development. Figure 9 represents the steps of this process.

Select participating simulations Once the required functionality is known, the simulation engineers must research candidate federates for inclusion in the federation. Some of the considerations for selection are the capability to model desired phenomena, ease of integration with the other models, and difficulty of required modifications. In some cases, a new federate must be developed in order to model some aspects of the system. For the PEO Soldier federation, the participating models are IWARS for the detailed representation of dismounted soldiers and a choice of Combat ^{XXI} or OneSAF for representations of supporting unmanned aircraft, vehicles, fires, and less detailed representations of OPFOR soldiers. The BCMS federate Organic Communication Service (OCS) generates spot reports based on detection events, the Message Transceiver Service (MTS) calculates propagation of radio messages, and the Situation Awareness and Display (SANDS) federate manages the local operating picture of each federate in the simulation. Finally, a fires federate and command and control federate will have to be developed in order to implement fires decisions and maneuver decisions based upon the situation awareness of small unit leaders.

Allocate simulation activities to specific simulation models Once the candidate federates are selected, modeling functions must be allocated to individual federates. The resulting functional allocation takes the functional flow diagram from the operational level and allocates specific functions to federates using swim lanes in the UML activity diagram. This allocation for the ground soldier command and control simulation is shown in figures 10 and 11. In a similar fashion, Figure 12 shows the allocation of command and control functions to simulation federates.

Allocate value measures to specific simulation models In a manner similar to the allocation of functions, the requirements to collect necessary performance data should be allocated to federates as well. In some cases, the required data may not exist in any one federate, but will have to be collected from network interaction data collected by loggers. In the case of the PEO Soldier federation, most of the required data was available within the network data logger used to support the experiment. However, a custom data logger was required to capture the knowledge level of friendly forces over the course of the simulation.

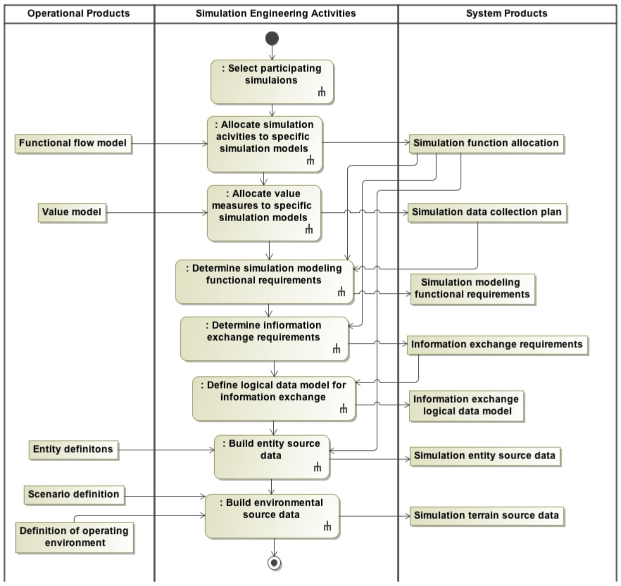

Operational Products
Simulation Engineering Activities
System Products
: Select participating simulaions
: Allocate simulation acivities to specific simulation models
Functional flow model
Simulation function allocation
: Allocate value measures to specific simulation models
Value model
Simulation data collection plan
: Determine simulation modeling functional requirements
Simulation modeling functional requirements
: Determine infiormation exchange requirements
Information exchange requirements
: Define logical data model for information exchange
Information exchange logical data model
: Build entity source data
Entity definitons
Simulation entity source data
: Build environmental source data
Scenario definition
Definition of operating environment
Simulation terrain source data

Figure 9: System level activities.

Define simulation modeling functional requirements Once the modeling functions and data collection functions have been determined, the simulation functionality requirements may be formally specified for the models. These requirements documents may be used to support contracting with federate developers who must deliver models with the required functions.

Determine information exchange requirements In order for the federation to execute, data must be exchanged between the models. These requirements may be derived from the activity diagrams used to allocate functions to individual federates. Any time that a control line crosses a swimlane, there is typically a requirement for some amount of information to be passed in order to support that allocation.

Define logical data model for information exchange As information exchange requirements are identified in the previous step, engineers must formally specify the data elements required to support that data exchange. These data requirements can often be specified in a UML class diagram. This is a two-way process. It may be more efficient to delay this formal specification until the information exchange architecture is selected in the technical view. In some cases, information elements from that architecture may be reverse engineered to provide the required information elements.

Build entity source data In addition to developing simulation software, the team must also consider the entities that will participate in the scenario. In some cases, these entities must be constructed from a significant amount of data. This step represents the collection of accurate and appropriate source data for the entities in the scenario.

Build environmental source data In addition to entities, the environment must be considered as well. This step represents the collection of source data necessary to appropriately represent the environment in the different federates. The environmental representation may not be the same for all federates. However, using the same source data will lead to correlated representations across the models. For the PEO Soldier federation, a terrain box at White Sands, New Mexico was used. A collection of geospatial elevation data, imagery, and shape files was compiled to support terrain data generation using a commercial tool.

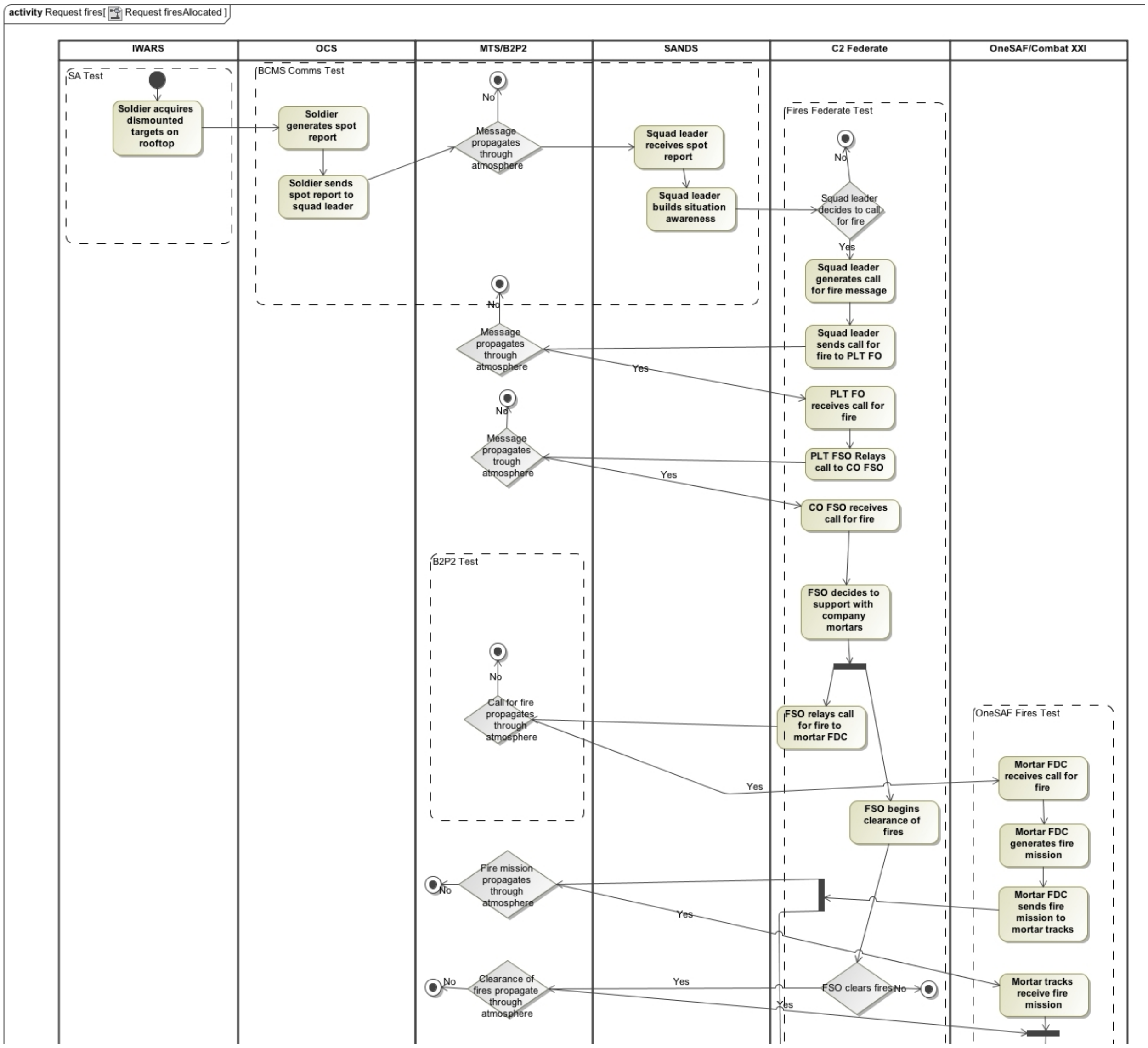

activity Request fires[Request firesAllocated]
IWARS
OCS
MTS/B2P2
SANDS
C2 Federate
OneSAF/Combat XXI
SA Test
Soldier acquires dismounted targets on rooftop
BCMS Comms Test
Soldier generates spot report
Soldier sends spot report to squad leader
No
Message propagates through atmosphere
Squad leader receives spot report
Squad leader builds situation awareness
Fires Federate Test
No
Squad leader decides to call for fire
Yes
Squad leader generates call for fire message
No
Message propagates through atmosphere
Squad leader sends call for fire to PLT FO
Yes
PLT FO receives call for fire
No
Message propagates trough atmosphere
PLT FSO Relays call to CO FSO
Yes
CO FSO receives call for fire
B2P2 Test
FSO decides to support with company mortars
No
Call for fire propagates through atmosphere
FSO relays call for fire to mortar FDC
OneSAF Fires Test
Yes
Mortar FDC receives call for fire
FSO begins clearance of fires
Mortar FDC generates fire mission
No
Fire mission propagates through atmosphere
Mortar FDC sends fire mission to mortar tracks
Yes
Mortar tracks receive fire mission
No
Clearance of fires propagate through atmosphere
Yes
FSO clears fires
No
Yes

Figure 10: Allocation of request fires functions to simulation federates, represented as vertical swim lanes (see continuation in Figure 11).

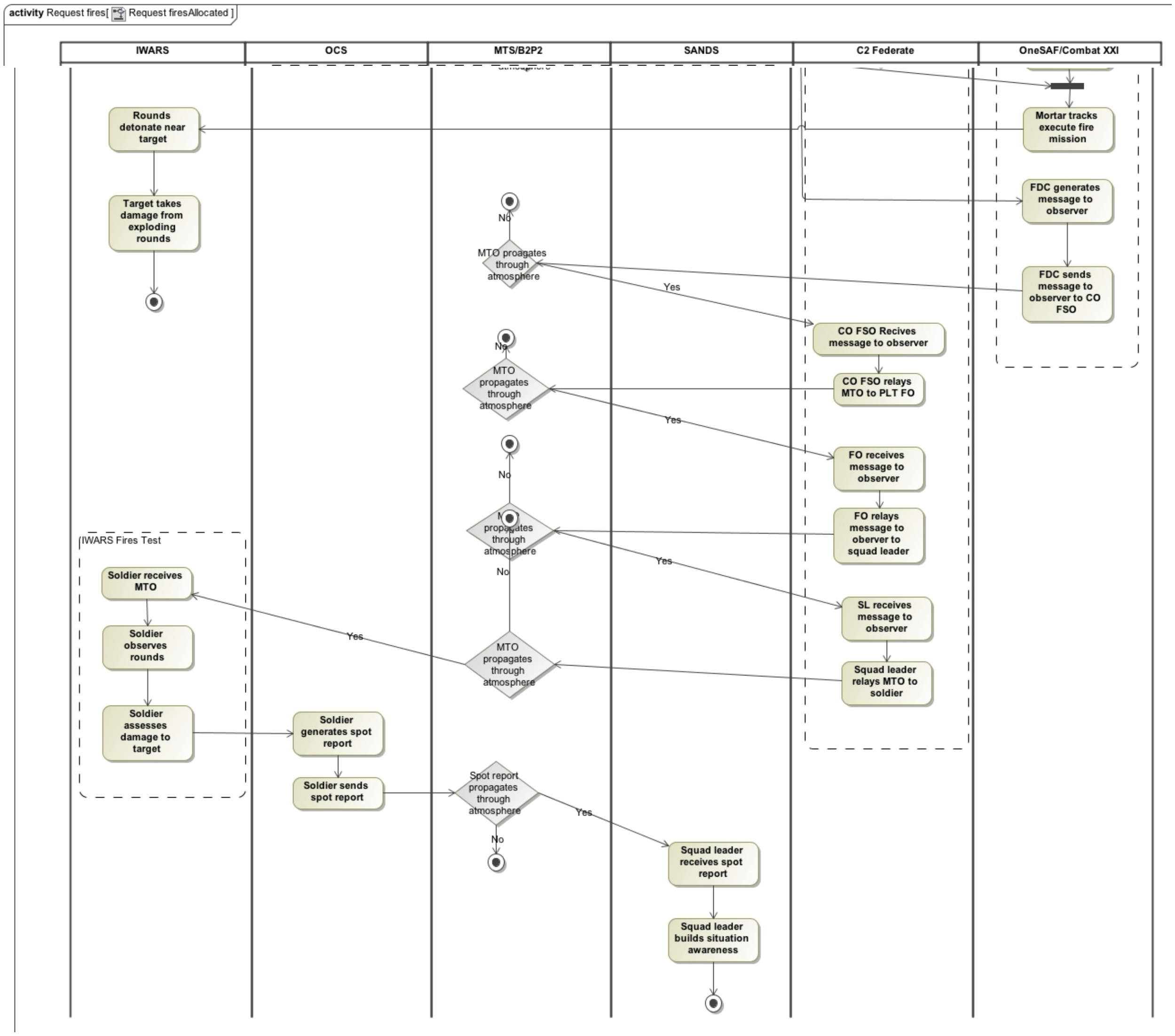

activity Request fires[Request firesAllocated]
IWARS
OCS
MTS/B2P2
SANDS
C2 Federate
OneSAF/Combat XXI
Mortar tracks execute fire mission
FDC generates message to observer
FDC sends message to observer to CO FSO
Rounds detonate near target
Target takes damage from exploding rounds
No
MTO proagates through atmosphere
Yes
CO FSO Recives message to observer
CO FSO relays MTO to PLT FO
No
MTO propagates through atmosphere
Yes
FO receives message to observer
FO relays message to oberver to squad leader
No
MTO propagates through atmosphere
Yes
No
SL receives message to observer
Squad leader relays MTO to soldier
IWARS Fires Test
Soldier receives MTO
Soldier observes rounds
Soldier assesses damage to target
Yes
MTO propagates through atmosphere
Soldier generates spot report
Soldier sends spot report
Spot report propagates through atmosphere
Yes
No
Squad leader receives spot report
Squad leader builds situation awareness

Figure 11: Allocation of request fires functions to simulation federates, represented as vertical swim lanes (continued from Figure 10).

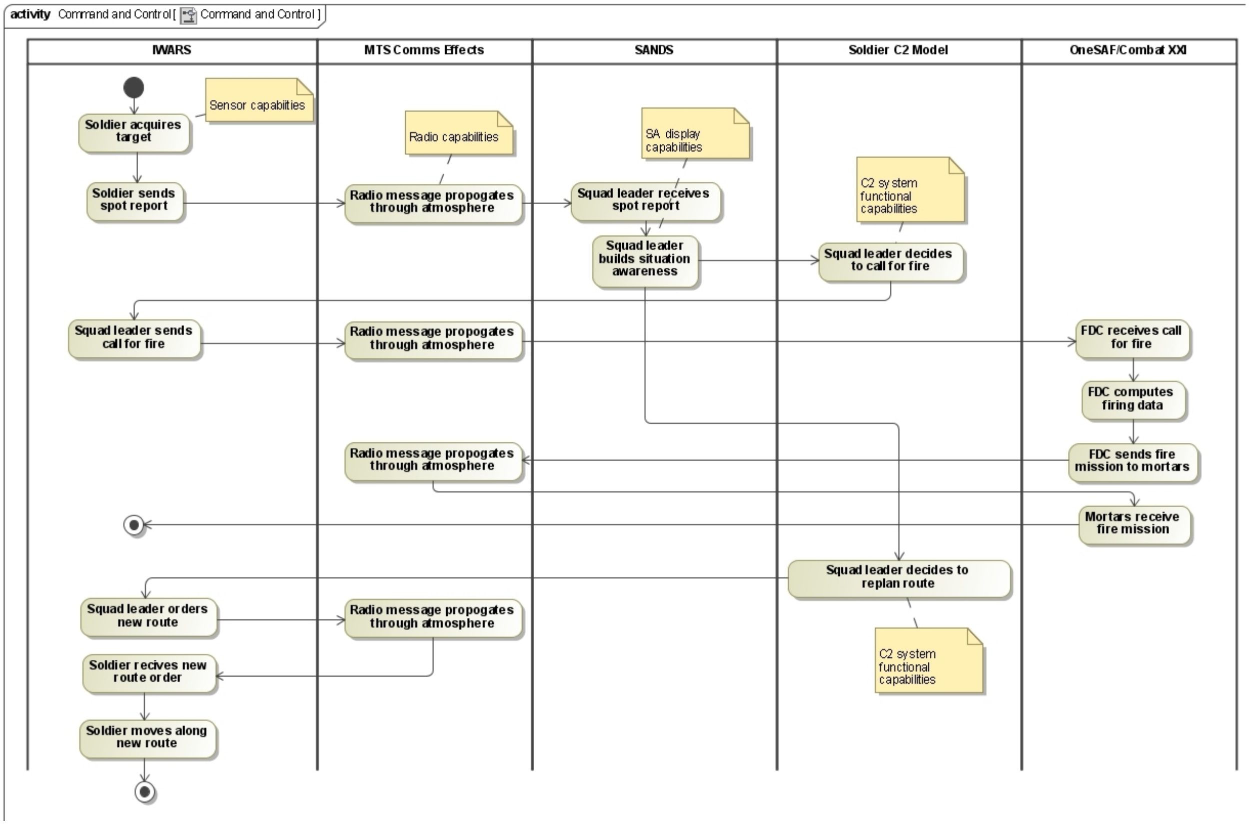

activity Command and Control [Command and Control]
IWARS
MTS Comms Effects
SANDS
Soldier C2 Model
OneSAF/Combat XXI
Sensor capabilities
Soldier acquires target
Soldier sends spot report
Radio capabilities
Radio message propogates through atmosphere
SA display capabilities
Squad leader receives spot report
Squad leader builds situation awareness
C2 system functional capabilities
Squad leader decides to call for fire
Squad leader sends call for fire
Radio message propogates through atmosphere
FDC receives call for fire
FDC computes firing data
Radio message propogates through atmosphere
FDC sends fire mission to mortars
Mortars receive fire mission
Squad leader decides to replan route
Squad leader orders new route
Radio message propogates through atmosphere
C2 system functional capabilities
Soldier recives new route order
Soldier moves along new route

Figure 12: Allocation of command and control functions to simulation federates.

3.3 Technical View Activities

Once the system has been designed and data has been collected, it is still necessary to do system development for all of the participating federates and for the overall federation integration architecture. These are all technical level activities that look to provide software engineers and programmers sufficient information that will allow them to write code and deliver working federates within the overall specification. Figure 13 shows a diagram of the technical level activities and products.

Select information exchange technical architecture The simulation must exchange information across an architecture designed for this purpose. Simulation standards such as Distributed Interactive Simulation (DIS) or the High Level Architecture (HLA) are possible choices. Another possibility is to use service oriented architectures, based on web standards, designed to support business or industrial systems. For the PEO Soldier federation, the MATREX High Level Architecture has been selected.

Develop information exchange data model The information exchange data model must be specified and represented in technical format selected in the previous step. In the case of HLA, this specification will be a federation object model (FOM). A web services architecture would require extensible markup language (XML) representations of the data. For the PEO Soldier federation, the MATREX FOM version 4.1 has been selected. This object model contains all of the necessary objects and interactions necessary to execute the command and control activities identified in the systems view. Once the information exchange data model is developed, a UML sequence diagram can be used to translate simulation functions into sequence diagrams that explicitly show the communications between federates and the simulation functions performed by each. Figures 14 through 18 show UML sequence diagrams to support the fires and command and control architectures identified in the systems view.

Specify simulation models Required simulation functions were determined as a systems level activity. Now these function must be specified using the language and format of the information exchange architecture. For example, in HLA, certain simulation functions could be started upon receipt of specific interactions from the run-time infrastructure (RTI). In a web services integration, these functions could be represented using the web services definition language (WSDL).

Build entity data models in simulation specific formats In this step, the entity data collected in the system level activity must be converted into input formats that can be read by the participating federates. These representations may be databases, spreadsheets, XML files, or other file formats required by the participating simulations. In some cases, supporting tools for the simulation can ease this transition. In other cases, it is a laborious manual process using basic editors. For the PEO Soldier federation, an XML file specified the entities and types required by each simulation, and each model built the necessary data files required to simulate each entity in their respective models.

Build data collection models in simulation specific formats In this step, the data collection requirements determined in the system level activity must be represented as output data from simulation federates or from data loggers tied to the federation. Depending upon the federate, these formats may be supported by standard database systems, or they may simply be text or log files that must be processed. The developers must build queries or tools that collect this raw data and summarize it as value measures from the value model built in the operational level. For the PEO Soldier federation, the most complicated data collection requirement was a capability to record the situation awareness state of each friendly entity for each minute of the battle. For this task, a custom MATREX federate was created to query SANDS each minute on behalf of each friendly entity. It wrote a data file including the known friendly and enemy positions on the battlefield.

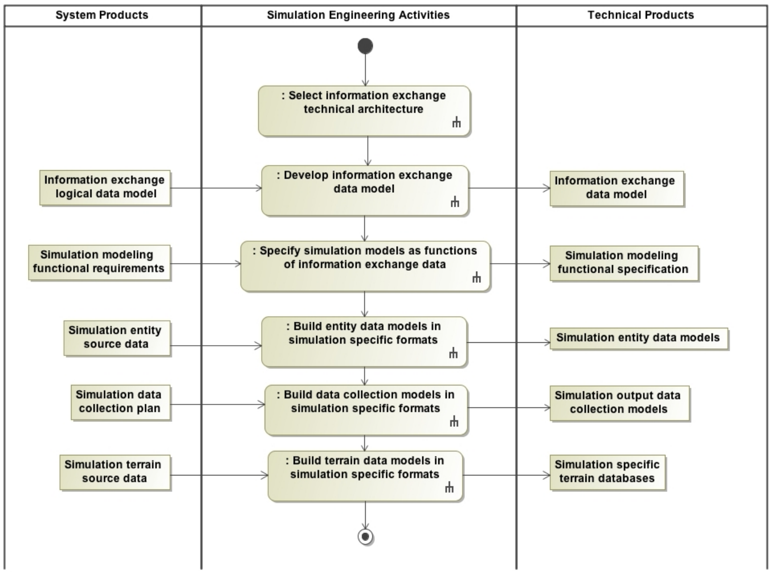

System Products
Simulation Engineering Activities
Technical Products
: Select information exchange technical architecture
Information exchange logical data model
: Develop information exchange data model
Information exchange data model
Simulation modeling functional requirements
: Specify simulation models as functions of information exchange data
Simulation modeling functional specification
Simulation entity source data
: Build entity data models in simulation specific formats
Simulation entity data models
Simulation data collection plan
: Build data collection models in simulation specific formats
Simulation output data collection models
Simulation terrain source data
: Build terrain data models in simulation specific formats
Simulation specific terrain databases

Figure 13: Technical level activities.

Build terrain data models in simulation specific formats

The source terrain data must also be converted into simulation specific formats. In many cases commercial tools support this process for a variety of formats. In other cases, the terrain data may be read into the simulation models in its existing geospatial format. The result of this step is correlated representation of the terrain across the federation. For the PEO Soldier federation, a commercial tool was used to generate correlated databases in the OneSAF ERC format and in an OpenFlight format for IWARS' 3-dimensional viewer.

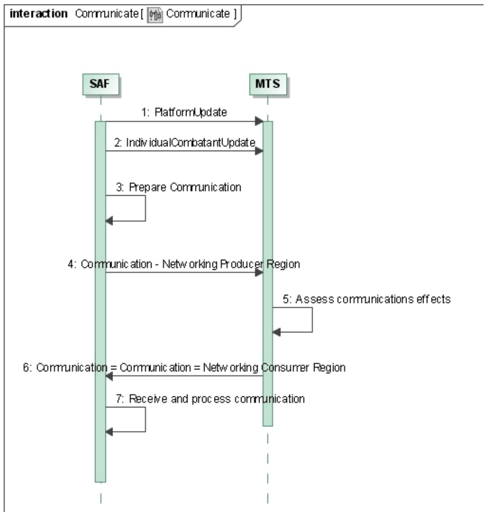

interaction Communicate [Communicate]
SAF
MTS
1: PlatformUpdate
2: Indiv idualCombatantUpdate
3: Prepare Communication
4: Communic ation - Netw orking Producer Region
5: Assess communications eff ects
6: Communication = Communic ation = Netw orking Consumer Region
7: Receive and proc ess communication

Figure 14: Communications sequence diagram. This diagram shows the sequence of messages required for semi-automated forces (SAF) entities in IWARS, OneSAF, and Combat ^{XXI} to send communications messages to each other. The MTS communications service subscribes to the platform updates of all entities and passes them along to the communications effects server. In this way, the server knows the locations, terrain types, and distances associated with point to point communications. SAF entities send MATREX communications using data distribution management (DDM) in the networking producer region. The MTS subscribes to this region, but the SAF simulations do not. The MTS will assess whether the message was delivered to its receivers. If so, it will publish that message using DDM in the networking consumer region. The SAF simulations will subscribe to this region and deliver the communications messages to the associated receiving entity for processing.

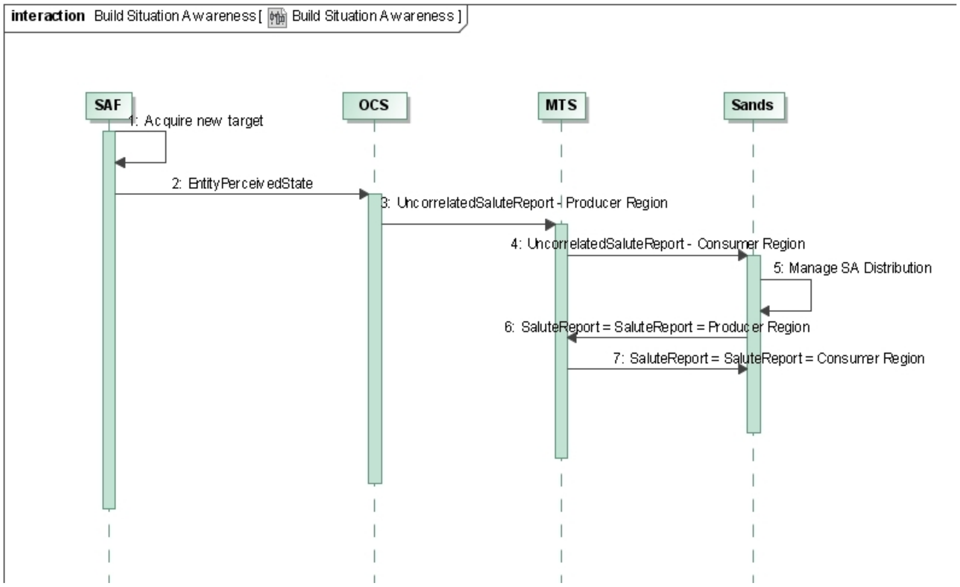

interaction Build Situation Awareness [Build Situation Awareness]
SAF
OCS
MTS
Sands
1: Acquire new target
2: EntityPerceivedState
3: UncorrelatedSaluteReport - Producer Region
4: UncorrelatedSaluteReport - Consumer Region
5: Manage SA Distribution
6: SaluteReport = SaluteReport = Producer Region
7: SaluteReport = SaluteReport = Consumer Region

Figure 15: Build situation awareness sequence diagram. This diagram shows the sequence of messages required for SAF entities in IWARS, OneSAF, and Combat ^{XXI} to build situation awareness for enemy forces. When a friendly entity on the battlefield acquires a target, it publishes a MATREX perception message. The OCS BCMS federate publishes a salute report after a delay period that would be required for the entity to build that report. This report is published on the networking producer region, assessed by the MTS for delivery, and if successful, published on the networking consumer region. The SANDS BCMS federate receives salute reports and updates the local operational picture (LOP) for the receiving entities. In addition, SANDS sends periodic spot reports up, down, and laterally in accordance with the communications hierarchy for friendly units. In this manner, SANDS maintains the current LOP of every friendly entity and makes it available to other federates via MATREX queries.

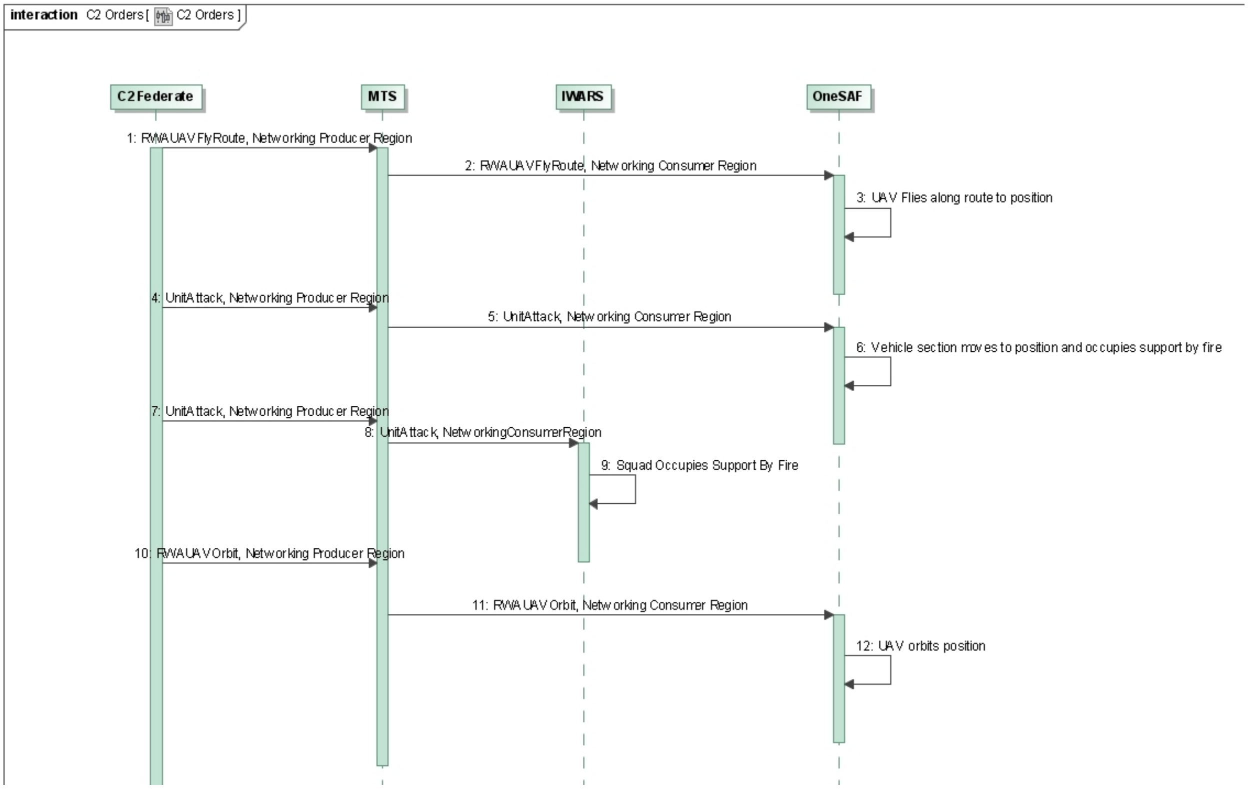

interaction C2 Orders [C2 Orders]
C2Federate
MTS
IWARS
OneSAF
1: RWAUAVFlyRoute, Networking Producer Region
2: RWAUAVFlyRoute, Networking Consumer Region
3: UAV Flies along route to position
4: UnitAttack, Networking Producer Region
5: UnitAttack, Networking Consumer Region
6: Vehicle section moves to position and occupies support by fire
7: UnitAttack, Networking Producer Region
8: UnitAttack, NetworkingConsumerRegion
9: Squad Occupies Support By Fire
10: RWAUAVOrbit, Networking Producer Region
11: RWAUAVOrbit, Networking Consumer Region
12: UAV orbits position

Figure 16: Command and control orders sequence diagram. This diagram shows the series of messages required to issue orders to IWARS, OneSAF, and Combat^{XXI} in order to execute an operations order. Each command is first issued using DDM on the networking producer region. The MTS federates determines whether the message can be delivered. If so, it is published on the networking consumer region for implementation by the receiving unit. Unmanned aircraft orders perform reconnaissance of the objective area, and unit movement orders move SAF entities to the objective area.

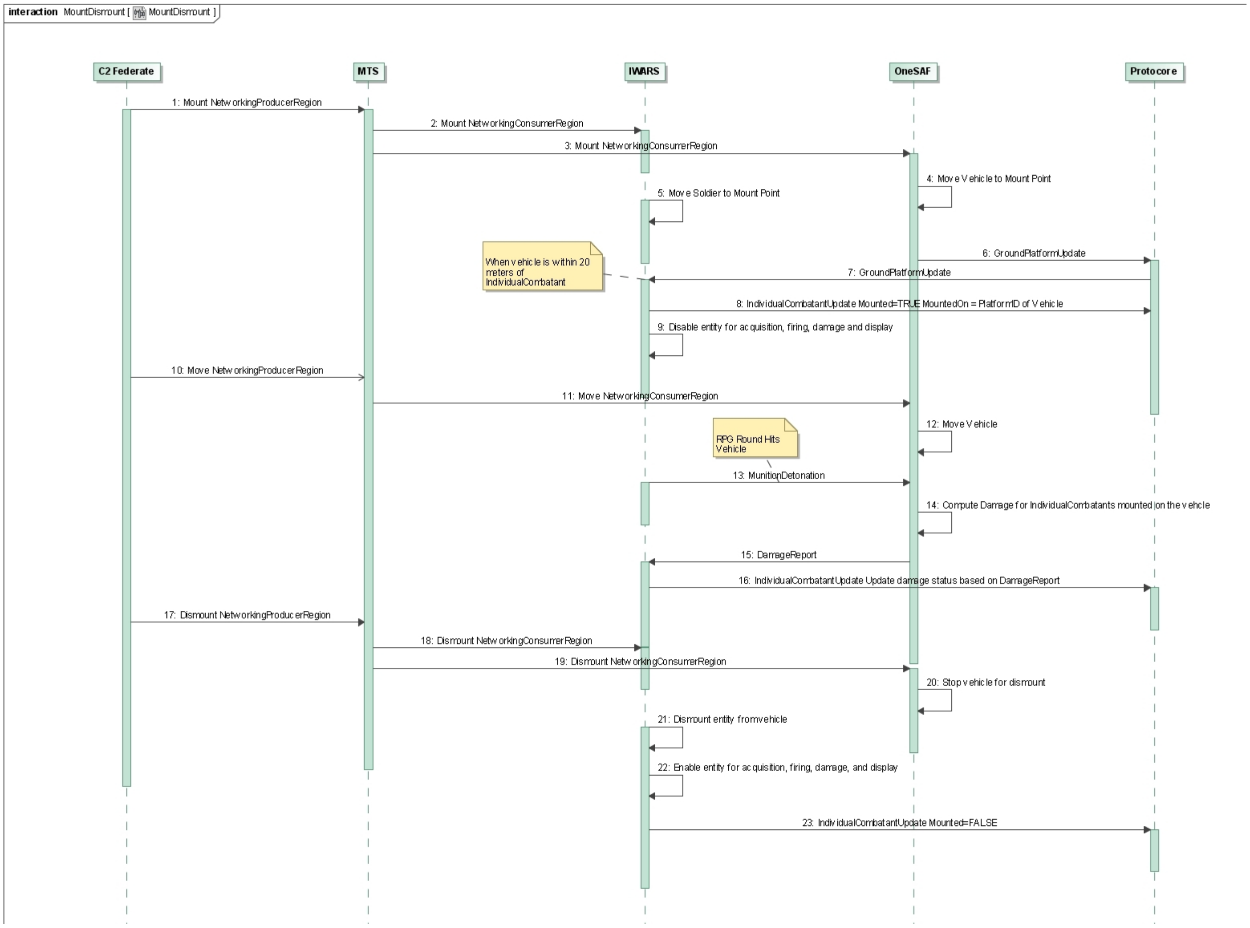

interaction MountDismount [MountDismount]
C2 Federate
MTS
IWARS
OneSAF
Protocore
1: Mount NetworkingProducerRegion
2: Mount NetworkingConsumerRegion
3: Mount NetworkingConsumerRegion
4: Move Vehicle to Mount Point
5: Move Soldier to Mount Point
6: GroundPlatformUpdate
When vehicle is within 20 meters of IndividualCombatant
7: GroundPlatformUpdate
8: IndividualCombatantUpdate Mounted=TRUE MountedOn = PlatformID of Vehicle
9: Disable entity for acquisition, firing, damage and display
10: Move NetworkingProducerRegion
11: Move NetworkingConsumerRegion
12: Move Vehicle
RPG Round Hits Vehicle
13: MunitionDetonation
14: Compute Damage for IndividualCombatants mounted on the vehcle
15: DamageReport
16: IndividualCombatantUpdate Update damage status based on DamageReport
17: Dismount NetworkingProducerRegion
18: Dismount NetworkingConsumerRegion
19: Dismount NetworkingConsumerRegion
20: Stop vehicle for dismount
21: Dismount entity fromvehicle
22: Enable entity for acquisition, firing, damage, and display
23: IndividualCombatantUpdate Mounted=FALSE

Figure 17: Mount-dismount sequence diagram. This diagram shows the sequence of messages required for an IWARS soldier to mount and dismount a OneSAF vehicle. First, the command and control federate issues a MATREX Mount interaction via the MTS communications service. Assuming both the soldier and vehicle receive the command, they both move to the mount point. Once the soldier is within 20 meters of the mount point, both simulations initiate the mounting process. IWARS disables the acquisition, engagement, damage, and display algorithms for the soldier. It publishes its state as mounted on the OneSAF vehicle. OneSAF records this and takes over managing the damage to soldiers mounted on its vehicles. As the OneSAF vehicle moves, a rocket propelled grenade engagement hits the vehicle in which the soldier is mounted. OneSAF runs its damage algorithm for mounted soldiers and publishes the results in a MATREX damage report message. Upon receiving the damage report, IWARS updates the damage state of the soldier to reflect the results of the OneSAF damage calculation. Finally, when the command and control federate issues a dismount command, OneSAF stops the vehicle. Once IWARS detects that the vehicle is stopped, it dismounts the entity, resuming internal algorithms for acquisition, firing, damage and display. IWARS then publishes updates for the soldier with the Mounted field set to FALSE.

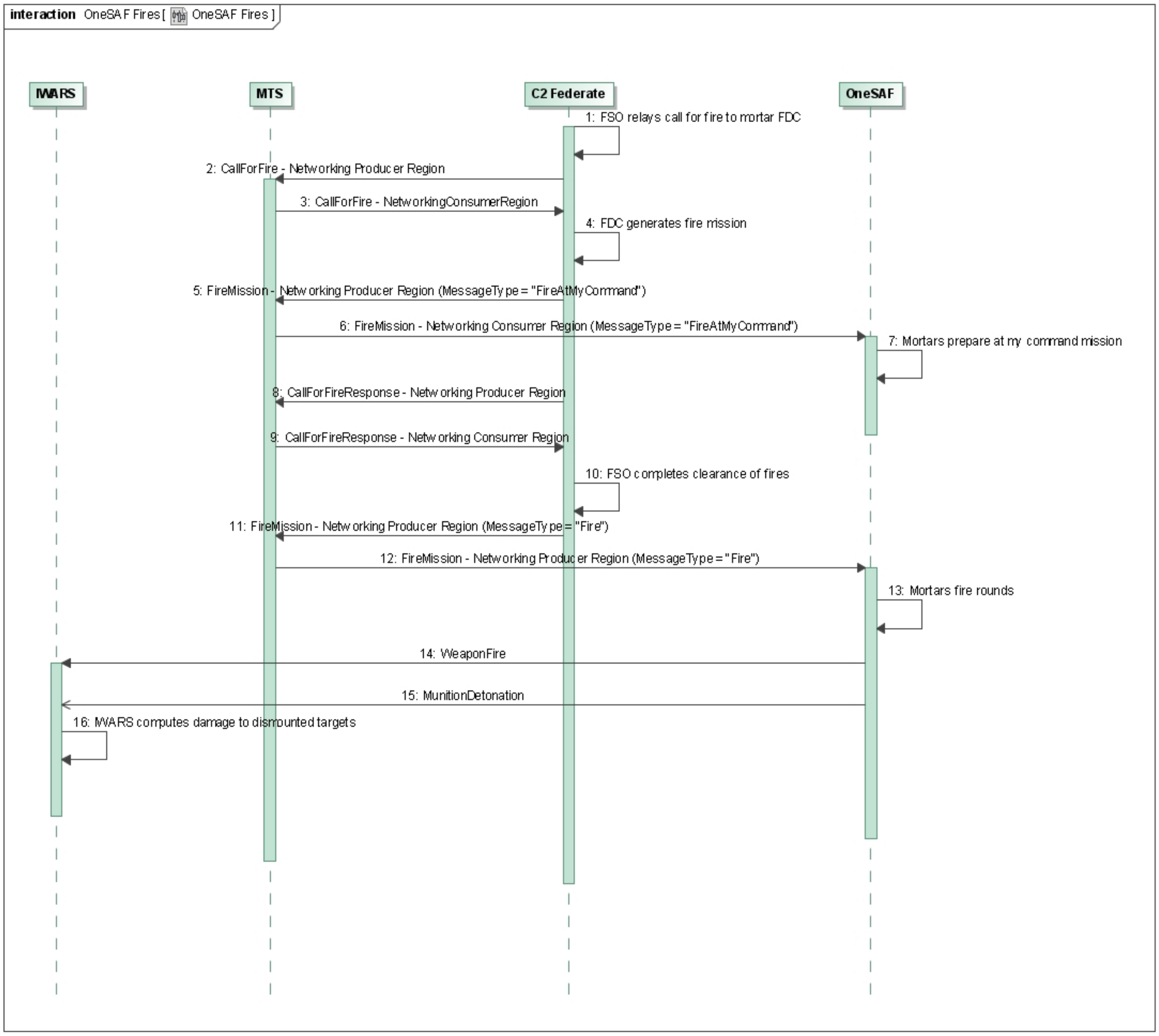

interaction OneSAF Fires [OneSAF Fires]
IWARS
MTS
C2 Federate
OneSAF
1: FSO relays call for fire to mortar FDC
2: CallForFire - Networking Producer Region
3: CallForFire - NetworkingConsumerRegion
4: FDC generates fire mission
5: FireMission - Networking Producer Region (MessageType = "FireAtMyCommand")
6: FireMission - Networking Consumer Region (MessageType = "FireAtMyCommand")
7: Mortars prepare at my command mission
8: CallForFireResponse - Networking Producer Region
9: CallForFireResponse - Networking Consumer Region
10: FSO completes clearance of fires
11: FireMission - Networking Producer Region (MessageType = "Fire")
12: FireMission - Networking Producer Region (MessageType = "Fire")
13: Mortars fire rounds
14: WeaponFire
15: MunitionDetonation
16: IWARS computes damage to dismounted targets

Figure 18: OneSAF fires sequence diagram. This diagram shows the sequence of messages required for OneSAF to fire mortar rounds in response to fire missions generated by the fires federate. The fires federate replicates the role of the fire direction center (FDC) that generates the fire mission and the fire support officer (FSO) who clears fires for the mission. First the FDC generates a fire mission in response to the call for fire message. The fire mission is passed via the MTS service to see if it can be delivered to the mortar unit. Once delivered, the mortars go through the process of laying the guns on the target and await a final fire message. In parallel with the process of laying the guns, the FSO conducts clearance of fires to ensure no friendly forces are in the vicinity of the mission. Once clearance of fires is complete, the FSO sends a fire message to the guns in order to execute the mission. The guns fire, and a munition detonation interaction is sent to IWARS so that damage can be calculated against dismounted targets in the area.

3.4 System Development and Testing

Delivery of the engineering products described at each level will give system developers all of the specifications they need to build software components that deliver the required functionality and interface with the federation architecture. The operational level will give them a conceptual context for integration. The system level will give them a semantic view of their components and an understanding of the overall simulation architecture. Finally the technical products will specify their components in sufficient detail to allow them to interface with the selected technical infrastructure. A good systems engineering process requires stakeholders from all three of these levels for the to work together in a coordinated way.

Despite the best intentions of a well designed architecture, there is no substitute for component and integration testing to ensure all of the pieces work as advertised. Component tests are designed around individual components and their interfaces. They typically test a discrete function by replicating the inputs from the federation and reading the outputs from the federate. The MATREX environment supported automatic test case generation to support integration. Larger scale integration tests bring together a number of federates and test the ability of the federation to model system-level capabilities and to collect system-level data. Annex C shows a sample MATREX test case generated to support the Mount/Dismount functionality for the federation. Simulation developers used similar test cases to support communications, situation awareness, fires, and command and control interactions. These test cases offered an unambiguous development specification for developers that minimized miscommunication and reduced the complexity of scheduled integration events.

3.5 Advantages of a Systems Engineering Approach

There are three main advantages to using a systems engineering approach to federated simulation development. The first advantage is to ensure a clear line of logic from operational representations, to system level federation design, to coding and development. A second advantage is the separation of concerns permitted by modeling the system on three different levels. Operational experts do not have to read computer code to adjust models on the operational level. System level experts can organize and specify the system using systems architecture tools, and code developers can work from technical level specifications. The final advantage is that all of the systems engineering products support the engineering manager in implementation of the development and test plan. It breaks the complex federation into discrete pieces of functionality that can be developed, component tested, and integration tested in order to manage progress. This approach has helped during implementation of the ground soldier command and control federation, saving a great deal of development time and effort that is typically spent rectifying poorly specified interfaces during integration tests.

4 Command and Control Federate Development

The systems view architecture work shown in Figures 10 through 12 reveals a requirement for a command and control federate to perform the fires and maneuver functions identified in the functional analysis. The West Point Operations Research Center took on the task of developing these MATREX federates for the PEO Soldier federation. The center developed the command and control federate to handle the maneuver decisions and the fires federate to handle the fires decisions.

The critical simulation role for these federates is to replicate doctrinal human decisions made during the course of the battle. These decisions require information, and the alternative ground soldier command and control architectures present different levels of situation awareness to the leaders making these decisions. In order to be effective, the fires federate must take situation awareness into account in order to perform fires decision tasks such as call for fire, fires approval and clearance of fires. The command and control federate must bring situation awareness into the maneuver decisions such as advancement of phases and selection of branches in the maneuver plan.

4.1 Fires Federate

Within Army doctrine, there are four key functions in the fires process. First, an observer must generate a call for fire based upon events on the battlefield (HQDA, 1991). If applicable, higher units will approve or disapprove the call for fire based upon the maneuver and fires plan. Next, the commander who owns the battlespace must clear fires within that space to ensure no friendly forces are in the area (HQDA, 2002). Once fires have been cleared, the fire support officer sends a fire mission to the company mortars for execution. Although this process seems pretty straightforward, it is very difficult to execute at the speed and distances required for support of small unit operations. Small unit trainers have developed a series of forward observer battle drills to better support small unit infantry operations with indirect fires (Pinnell & Ivanoff, 2003). Even with these battle drills, the fog of the battle and limited communications channels hamper execution, especially with respect to situation awareness. It is possible for a ground soldier command and control system to augment the capability of a small unit to execute these fires (Copeland, 2007a).

Figures 10 and 11 show the high level design requirements for the fires federate. The fires federate is required to capture the differences in the fires process given different command and control system capabilities. Figure 19 shows this process as it is implemented at platoon level within the fires federate. Consider two differently equipped platoons. The Rapid Fielding Initiative (RFI) platoon is equipped as current infantry platoons with voice radios down to squad level used for command and control. The Ground Soldier System (GSS) platoon is equipped with the Ground Soldier System command and control capability down to the squad leader. The system employs the following capabilities:

- Automatic passing and plotting of enemy situation awareness (SALUTE reports)
- Automatic passing and plotting of friendly situation awareness (Situation reports)
- Generation of call for fire messages from enemy situation awareness picture
- Clearance of fires decision support based upon target location and munition in the call for fire
- Fires approval decision support based upon clearance of fires and fire support plan
- Call for fire forwarding to next echelon in fires process

The capabilities enable a more accurate and expeditious fires process for the dismounted platoon. Figures 20 through 22 highlight these differences.

The technical implementation of the fires federate required the capture of data related to the fires process. The West Point development team implemented this data in the extensible markup language (XML). The key reference data elements and task organization elements for the mission are as follows:

Call for fire equipment These are the capabilities of the equipment used by a dismounted soldier to call for fire. These capabilities relate to the time required to request, receive and understand enemy situation awareness and the time required to generate and send a call for fire mission.

Fires approval equipment These are the capabilities of the equipment used by a dismounted soldier to approve fires.

Fire support equipment These are the capabilities of the equipment used by the fire support officer to approve fires and generate a fire mission.

Battlefield area This is a specific region of the battlefield within which certain fires planning factors and priorities apply.

Attack guidance matrix This matrix identifies the types of targets to be engaged by direct fire in priority. Each target is considered a cluster of targets within a certain distance of each other. For example, in order to prevent fires at individual enemy dismounts, the fire support planner may

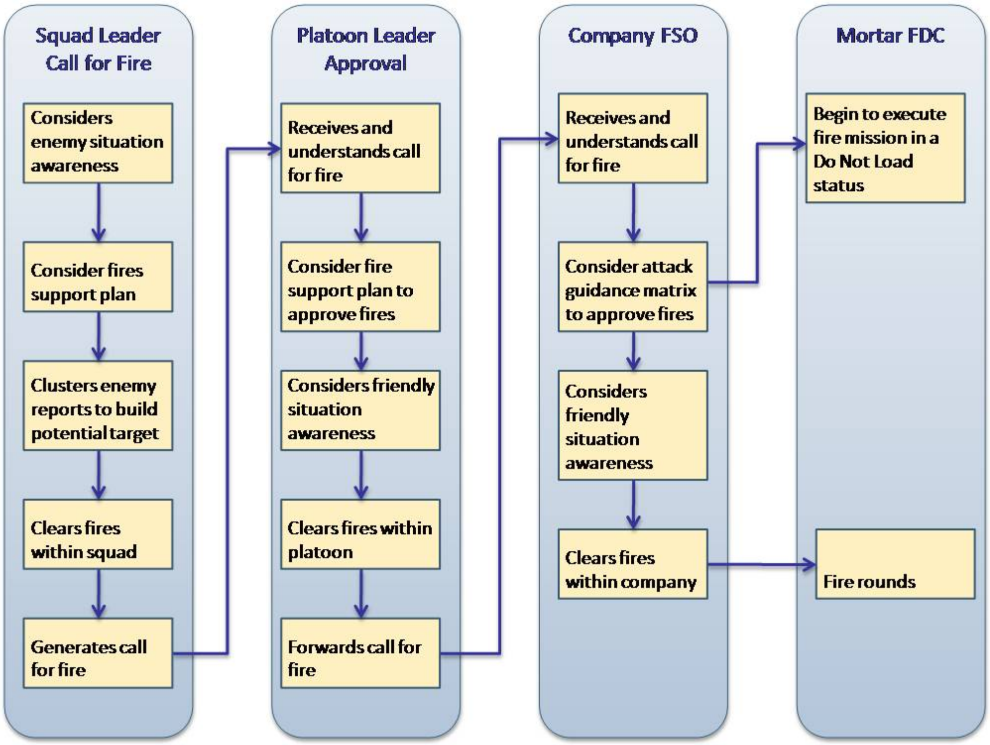
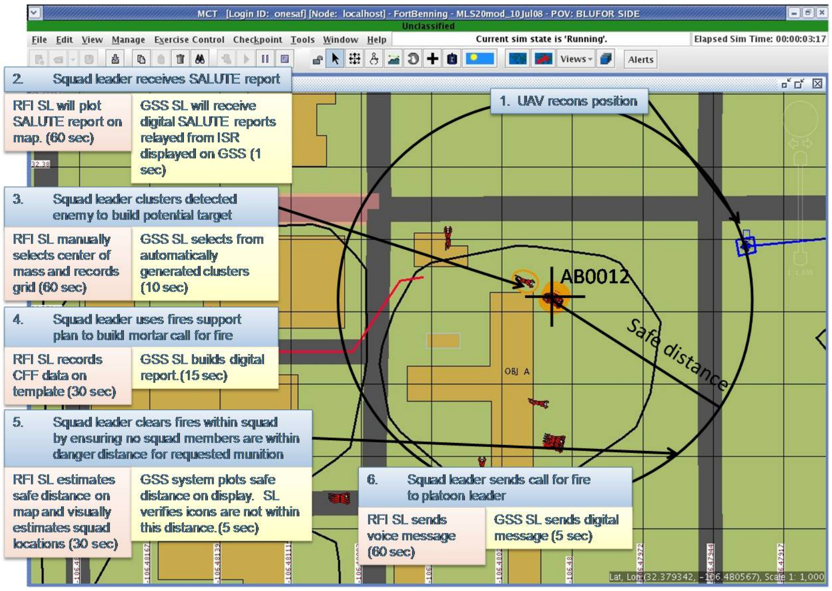

Squad Leader Call for Fire
Considers enemy situation awareness
Consider fires support plan
Clusters enemy reports to build potential target
Clears fires within squad
Generates call for fire
Platoon Leader Approval
Receives and understands call for fire
Consider fire support plan to approve fires
Considers friendly situation awareness
Clears fires within platoon
Forwards call for fire
Company FSO
Receives and understands call for fire
Consider attack guidance matrix to approve fires
Considers friendly situation awareness
Clears fires within company
Mortar FDC
Begin to execute fire mission in a Do Not Load status
Fire rounds

Figure 19: The fires process as modeled by the Fires Federate.


MCT [Login ID: onesaf] [Node: localhost] - FortBenning - MLS20mod_10Jul08 - POV: BLUFOR SIDE
Unclassified
File Edit View Manage Exercise Control Checkpoint Tools Window Help
Current sim state is 'Running'.
Elapsed Sim Time: 00:00:03:17
Views
Alerts
AB0012
OBJ A
Safe distance
Lat, Lon (32.379342, -106.480567), Scale 1: 1,000

1. UAV recons position

2. Squad leader receives SALUTE report

RFI SL will plot SALUTE report on map. (60 sec)

GSS SL will receive digital SALUTE reports relayed from ISR displayed on GSS (1 sec)

3. Squad leader clusters detected enemy to build potential target

RFI SL manually selects center of mass and records grid (60 sec)

GSS SL selects from automatically generated clusters (10 sec)

4. Squad leader uses fires support plan to build mortar call for fire

RFI SL records CFF data on template (30 sec)

GSS SL builds digital report.(15 sec)

5. Squad leader clears fires within squad by ensuring no squad members are within danger distance for requested munition

RFI SL estimates safe distance on map and visually estimates squad locations (30 sec)

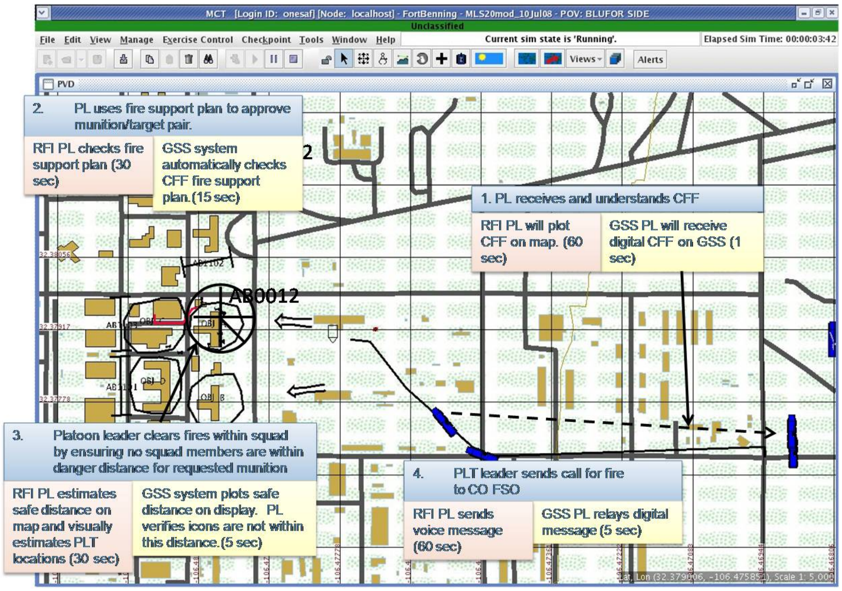
GSS system plots safe distance on display. SL verifies icons are not within this distance.(5 sec)

6. Squad leader sends call for fire to platoon leader

RFI SL sends voice message (60 sec)

GSS SL sends digital message (5 sec)

Figure 20: Squad leader calls for fire. This diagram shows the time and process differences between an RFI-equipped squad leader and a GSS-equipped squad leader for the call for fire task.


MCT [Login ID: onesaf] [Node: localhost] - FortBenning - MLS20mod_10Jul08 - POV: BLUFOR SIDE
Unclassified
File Edit View Manage Exercise Control Checkpoint Tools Window Help
Current sim state is 'Running'.
Elapsed Sim Time: 00:00:03:42
Views
Alerts
PVD
2
AB0012
OBJ
OBJ B
Lat, Lon (32.379006, -106.475851), Scale 1: 5,000

1. PL receives and understands CFF

RFI PL will plot CFF on map. (60 sec)

GSS PL will receive digital CFF on GSS (1 sec)

2. PL uses fire support plan to approve munition/target pair.

RFI PL checks fire support plan (30 sec)

GSS system automatically checks CFF fire support plan.(15 sec)

3. Platoon leader clears fires within squad by ensuring no squad members are within danger distance for requested munition

RFI PL estimates safe distance on map and visually estimates PLT locations (30 sec)

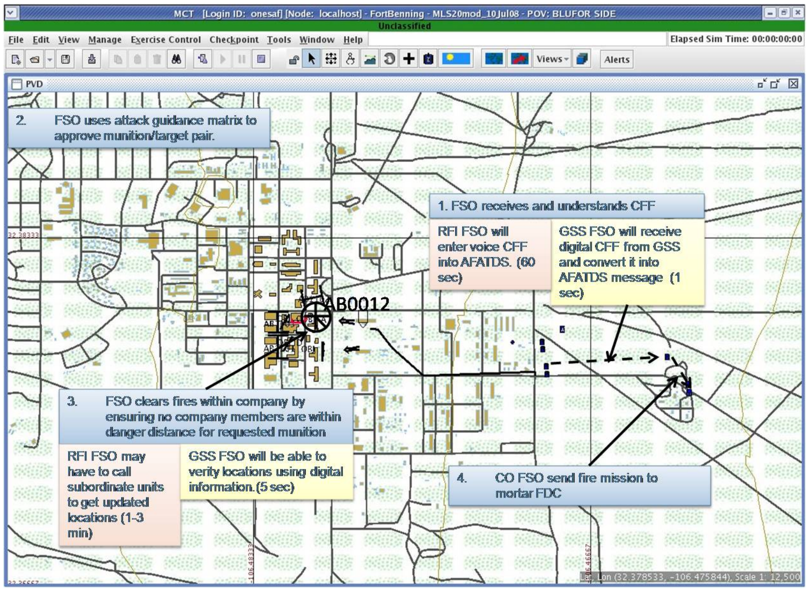
GSS system plots safe distance on display. PL verifies icons are not within this distance.(5 sec)

4. PLT leader sends call for fire to CO FSO

RFI PL sends voice message (60 sec)

GSS PL relays digital message (5 sec)

Figure 21: Platoon leader approves fires. This diagram shows the time and process differences between an RFI-equipped squad leader and a GSS-equipped platoon leader for the approve and forward call for fire task.


MCT [Login ID: onesaf] [Node: localhost] - FortBenning - MLS20mod_10Jul08 - POV: BLUFOR SIDE
Unclassified
File Edit View Manage Exercise Control Checkpoint Tools Window Help
Elapsed Sim Time: 00:00:00:00
Views
Alerts
PVD
AB0012

1. FSO receives and understands CFF

RFI FSO will enter voice CFF into AFATDS. (60 sec)

GSS FSO will receive digital CFF from GSS and convert it into AFATDS message (1 sec)

2. FSO uses attack guidance matrix to approve munition/target pair.

3. FSO clears fires within company by ensuring no company members are within danger distance for requested munition

RFI FSO may have to call subordinate units to get updated locations (1-3 min)

GSS FSO will be able to verity locations using digital information.(5 sec)

4. CO FSO send fire mission to mortar FDC

Figure 22: Fire support officer generates fire mission. This diagram shows the time and process differences that occur when the fire support officer receives a call for fire from an RFI-equipped platoon as opposed to a GSS-equipped platoon.

list five enemy dismounts within a 50 meter radius of each other as a target. This will cause units to call for fire only on groups of enemy dismounts of five or more. Each target also includes a planned munition, number of rounds, and fuze type, along with a minimum safe distance for friendly forces given that munition and fuze type.

Priorities of fire This is a list of friendly units in order of priority. If multiple calls for fire are processing at once, those from higher priority units will be processed first.

Clearance of fire rules The first element of these rules is a time threshold for the age of friendly situation reports. If the situation report from any friendly unit is older than this threshold, clearance of fires will not be granted until a new situation report is received and all units are verified as outside of the minimum safe distance for the mission. In addition, these rules allow the specification of robotic units, such as unmanned aircraft systems, which may be placed at risk. Fires will be cleared even if these units are within the minimum safe distance for the mission.

Call for fire units These are units that are capable of generating a call for fire, such as a squad leader. Each unit will is further defined by its call for fire equipment, the attack guidance matrix used to generate targets, and the unit to which the call for fire message will be sent.

Call for fire approval units These are units that perform the role of approving calls for fire. They will use specific fires approval equipment in this role. They will consider priorities of fire, the attack guidance matrix, and clearance of fires rules in this process. They will forward the approval to other units in accordance with their defined relay role.

Fire support officers These are leaders that accept calls for fire, clear and approve fires, and pass fire missions to firing units for execution.

Firing units These are mortar and artillery firing units that have certain munitions available to fire at targets.

Figures 23 and 24 show a graphical depiction of the XML structure of the data required by the fires federate.

The West Point development team selected the Scala programming language for implementation of the fires federate (Odersky *et al.* , 2006). The key factors for its selection are its support for functional programming, java interoperability, polymorphism, multiple inheritance, multiple threads, and XML processing. The fires federate runs as a Protocore MATREX federate. It queries the BCMS SANDS federate for situation awareness in the execution of the fires process. In this manner, the resulting fires are based upon a simulation of the situation awareness and fires processing capabilities of the unit's command and control equipment.

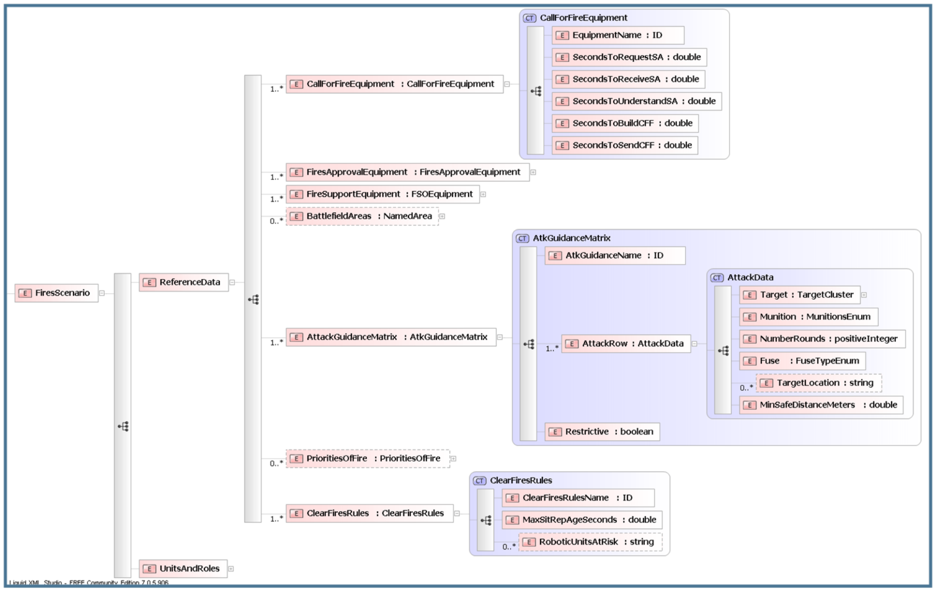

FiresScenario
ReferenceData
UnitsAndRoles
1..* CallForFireEquipment : CallForFireEquipment
CallForFireEquipment
EquipmentName : ID
SecondsToRequestSA : double
SecondsToReceiveSA : double
SecondsToUnderstandSA : double
SecondsToBuildCFF : double
SecondsToSendCFF : double
1..* FiresApprovalEquipment : FiresApprovalEquipment
1..* FireSupportEquipment : FSOEquipment
0..* BattlefieldAreas : NamedArea
1..* AttackGuidanceMatrix : AtkGuidanceMatrix
AtkGuidanceMatrix
AtkGuidanceName : ID
1..* AttackRow : AttackData
AttackData
Target : TargetCluster
Munition : MunitionsEnum
NumberRounds : positiveInteger
Fuse : FuseTypeEnum
0..* TargetLocation : string
MinSafeDistanceMeters : double
Restrictive : boolean
0..* PrioritiesOfFire : PrioritiesOfFire
1..* ClearFiresRules : ClearFiresRules
ClearFiresRules
ClearFiresRulesName : ID
MaxSitRepAgeSeconds : double
0..* RoboticUnitsAtRisk : string
Liquid XML Studio - FREE Community Edition 7.0.5.906

Figure 23: Fires data elements used by the Fires Federate.

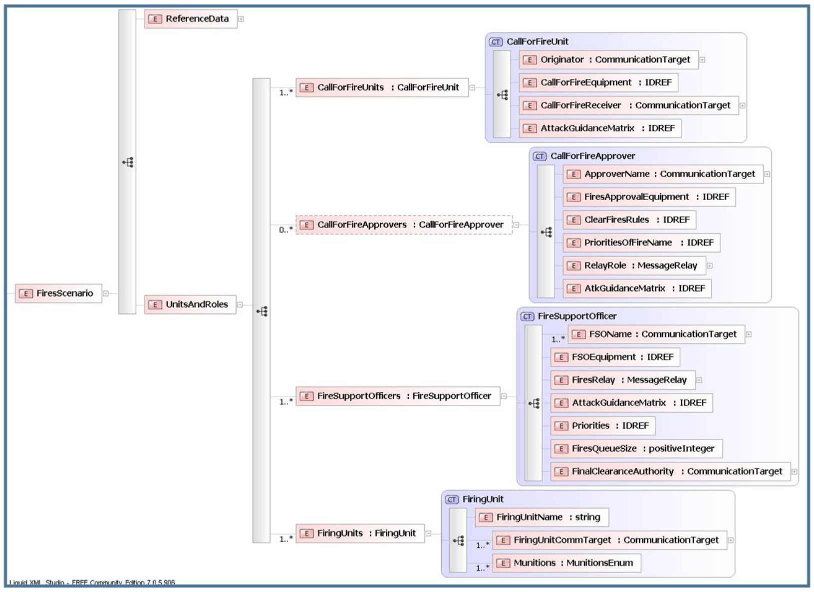

FiresScenario
ReferenceData
UnitsAndRoles
1..* CallForFireUnits : CallForFireUnit
CallForFireUnit
Originator : CommunicationTarget
CallForFireEquipment : IDREF
CallForFireReceiver : CommunicationTarget
AttackGuidanceMatrix : IDREF
0..* CallForFireApprovers : CallForFireApprover
CallForFireApprover
ApproverName : CommunicationTarget
FiresApprovalEquipment : IDREF
ClearFiresRules : IDREF
PrioritiesOfFireName : IDREF
RelayRole : MessageRelay
AtkGuidanceMatrix : IDREF
1..* FireSupportOfficers : FireSupportOfficer
FireSupportOfficer
1..* FSOName : CommunicationTarget
FSOEquipment : IDREF
FiresRelay : MessageRelay
AttackGuidanceMatrix : IDREF
Priorities : IDREF
FiresQueueSize : positiveInteger
FinalClearanceAuthority : CommunicationTarget
1..* FiringUnits : FiringUnit
FiringUnit
FiringUnitName : string
1..* FiringUnitCommTarget : CommunicationTarget
1..* Munitions : MunitionsEnum
Liquid XML Studio - FREE Community Edition 7.0.5.906

Figure 24: Fire units and roles used by the Fires Federate.

4.2 Command and Control Federate

There are a few high-level design requirements for the command and control federate, as illustrated in Figure 12. First, it must base phasing and branching of based upon situation awareness. An information requirement can be thought of as a yes or no question about the tactical scenario using a leader's awareness of friendly and enemy forces. The operation moves to next maneuver phase based upon satisfying information requirements. Different branches of a phase are selected based upon satisfying information requirements. Second, the federate must be able to issue commands to all participating federates. In other words, a movement order issued by the command and control federate is exactly the same, regardless of whether the executing squad is owned by IWARS, OneSAF, or Combat^{XXI}. This requires development in each participating simulation to respond to these commands.

The introduction of digital battle command to the dismounted soldier via the Land Warrior system has greatly increased the dismounted leader's situation awareness picture and ability to control maneuver of his elements (Copeland, 2007b). The Ground Soldier System (GSS) looks to further increase these capabilities with improved hardware and integration with other forces. If a GSS soldier has better situation awareness than an RFI soldier, this will lead to more timely and correct maneuver decisions.

Figure 25 represents the command and control process for the scenario executed by the PEO Soldier federation. This process is completely dependent upon the passing of orders and messages on the command and control network. For example, the platoon leader will not order the dismounted assault of the OBJ A to begin until he is aware that Section A has reached SBF A1. He will only reach this awareness if Section A reports its location via the command and control network. In other words, the sequencing of operations is based upon friendly situation awareness. He issues this command to 1st Squad on the command and control network. Next, the platoon leader will base the decision to use Axis North or Axis South based upon the presence of enemy forces in NAI 1. The only asset that has good visibility of NAI 1 is UAV1. Therefore, he will only know about enemy forces in NAI 1 is he receives a SALUTE report via the command and control network. In this case, his maneuver decision is based upon enemy situation awareness. By this design, the federation models command and control as a series of messages passed on a communications network that build situation awareness and order execution of tactical tasks based on that awareness.

The technical implementation of the command and control federate required the capture of data related to the command and control process. The West Point development team implemented this data in the extensible markup language (XML). The key reference data elements and task organization elements for the mission are as follows:

Information requirement All decisions made by the federate are based upon the information requirement. This is a yes or no question about the tactical situation that can be answered based upon the leaders awareness of friendly forces, enemy forces, and the terrain. An information requirement data structure consists of an identification of the deciding leader, a tactical graphic that represents an area of the battlefield, and a query of friendly or enemy forces related to that tactical graphic.

Proximity requirement This data structure defines a specific area of the battlefield that will be queried for the presence of friendly or enemy forces. It may be proximity specification with a point and a specific radius, or a containment specification that consists of a polygon in which forces may be contained.

Friendly unit requirement There are three ways to query situation awareness for an information requirement. The friendly unit requirement asks whether a certain number of entities from a specific unit are present in a certain area.

Friendly platform requirement This requirement asks whether a certain friendly platform is present in a certain area.

Enemy platform requirement This requirement asks whether a certain number of enemy platforms of

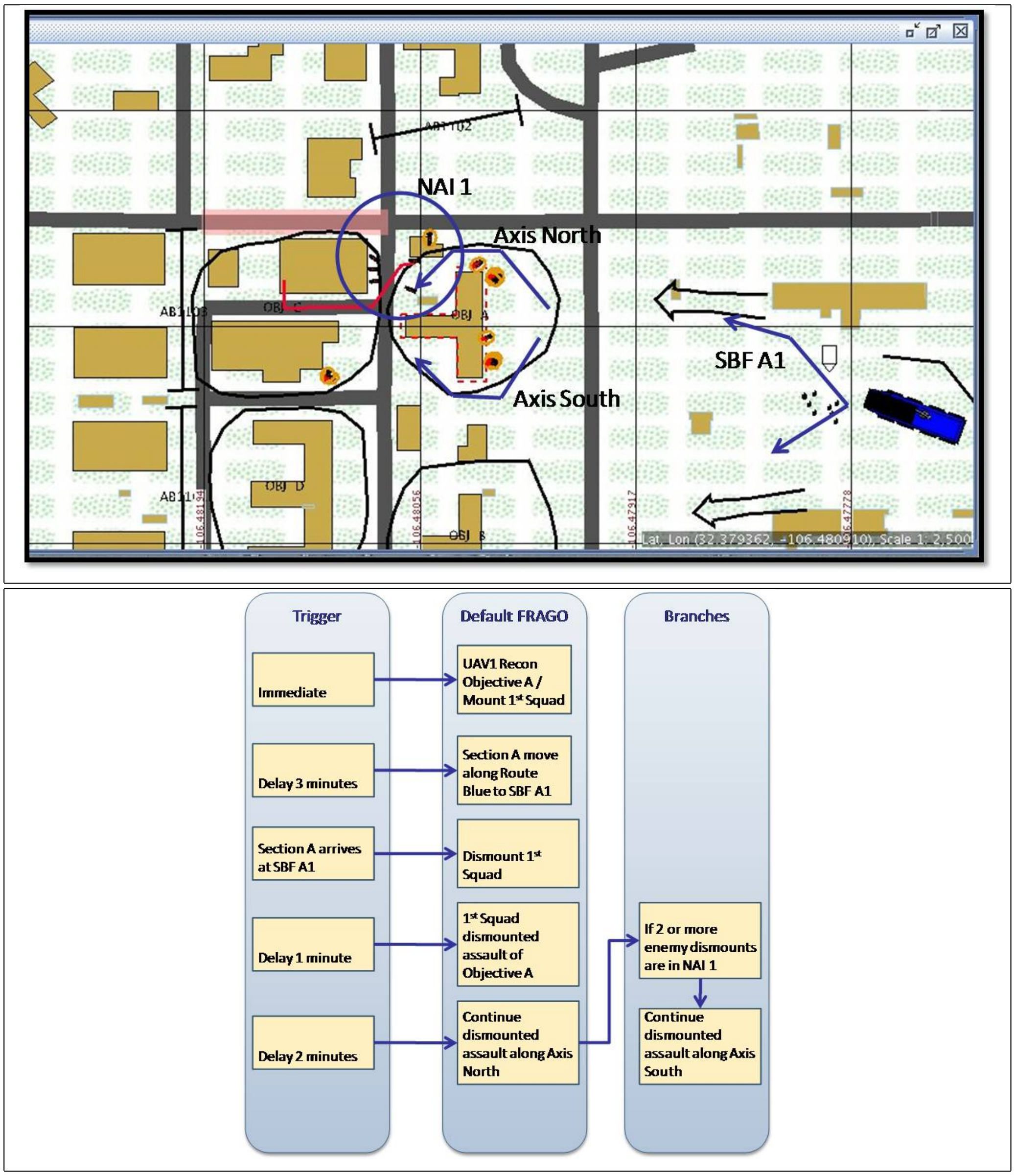

AB1102
NAI 1
Axis North
AB1103
OBJ C
OBJ A
SBF A1
Axis South
AB1104
OBJ D
OBJ B
-106.48194
-106.48056
-106.47917
-106.47778
Lat, Lon (32.379362, -106.480910), Scale 1: 2,500
Trigger
Default FRAGO
Branches
Immediate
UAV1 Recon Objective A / Mount 1st Squad
Delay 3 minutes
Section A move along Route Blue to SBF A1
Section A arrives at SBF A1
Dismount 1st Squad
Delay 1 minute
1st Squad dismounted assault of Objective A
If 2 or more enemy dismounts are in NAI 1
Delay 2 minutes
Continue dismounted assault along Axis North
Continue dismounted assault along Axis South

Figure 25: Scenario graphic illustrating command and control federate execution. Immediately upon execution of the mission, the platoon leader orders 1st squad to mount vehicles and the unmanned aircraft (UAV1) to conduct reconnaissance of the objective. The platoon waits 3 minutes for UAV1 to get to the objective before ordering Section A, consisting of 2 Bradley Fighting Vehicles (BFV), to move along Route Blue to its support by fire position (SBF A1). Once Section A reports they are in SBF A1, the platoon leader orders 1st Squad to dismount. He waits one minute and orders them to conduct a dismounted assault of Objective A (OBJ A). As the squad reaches OBJ A, there are two possible routes around the building in the objective area. The default route is Axis North. However, the platoon leader would like to avoid contact if possible. If there are two or more enemy dismounts in Named Area of Interest 1 (NAI 1), the platoon leader will order first squad to continue the attack along Axis South.

a certain type are present in an area of the battlefield.

Mission A mission is a named tactical operation that consists of one or more phases.

Phase A phase is a numbered and named component of a mission that consists of a trigger condition and a default set of tactical orders to be executed by units. It may also contain a series of branch orders to be executed is the associated information requirement is met.

Phase trigger This is a condition that must be met in order to execute a phase. A phase can execute immediately, after a delay in seconds, or wait until an information requirement is satisfied.

FRAGO This is a fragmentary order, a set of tactical tasks to be executed by units. The orders implemented by the PEO Soldier federation include movement, mounting and dismounting of forces, a unit attack, and route and orbit missions for unmanned aircraft.

Branch A branch is an alternate FRAGO to be executed if a particular information requirement is satisfied. A phase can contain one or more branches. The branches are queried on order of priority. If the information requirement for the first branch is met, its FRAGO will be executed. If not, the deciding leader will check to see of the information requirement for the second branch is met. This continues for all branches. If the information requirements for all branches are not satisfied, the deciding leader will issue the default FRAGO for that phase.

Figures 26 and 27 show a graphical depiction of the XML structure of the data required by the command and control federate.

The West Point development team implemented the command and control federate in the Scala programming language. The resulting Protocore MATREX federate queried the BCMS SANDS federate for situation awareness and issued tactical orders using the Command interaction in the MATREX FOM. With this implementation, leaders built situation awareness in the MATREX BCMS architecture ensuring that network topologies and communications effects are considered for each entity. In addition, leaders issued orders across that same communications network based upon their own situation awareness. The resulting federate allowed the federation to capture the decision making delays and effects that are introduced by different levels of situation awareness.

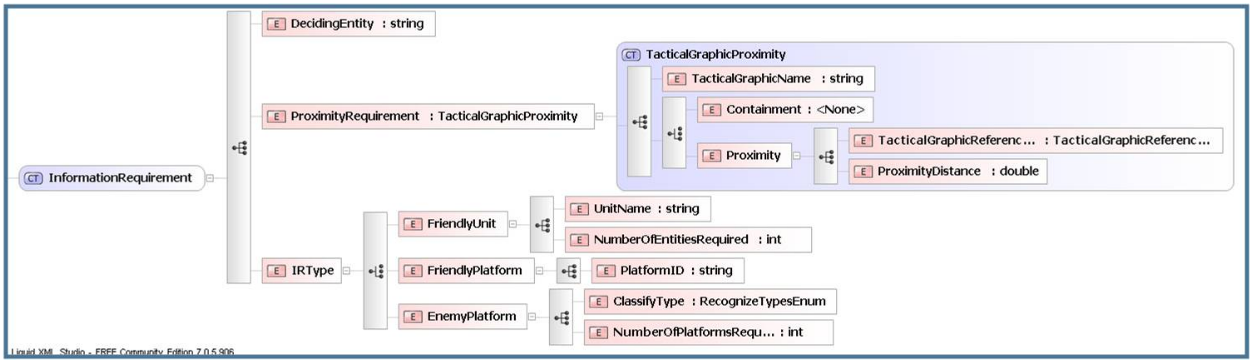

E DecidingEntity : string
CT TacticalGraphicProximity
E TacticalGraphicName : string
E ProximityRequirement : TacticalGraphicProximity
E Containment : <None>
E TacticalGraphicReferenc... : TacticalGraphicReferenc...
E Proximity
E ProximityDistance : double
CT InformationRequirement
E UnitName : string
E FriendlyUnit
E NumberOfEntitiesRequired : int
E IRType
E FriendlyPlatform
E PlatformID : string
E ClassifyType : RecognizeTypesEnum
E EnemyPlatform
E NumberOfPlatformsRequ... : int
Liquid XML Studio - FREE Community Edition 7.0.5.906

Figure 26: Information requirement data elements used by the command and control federate.

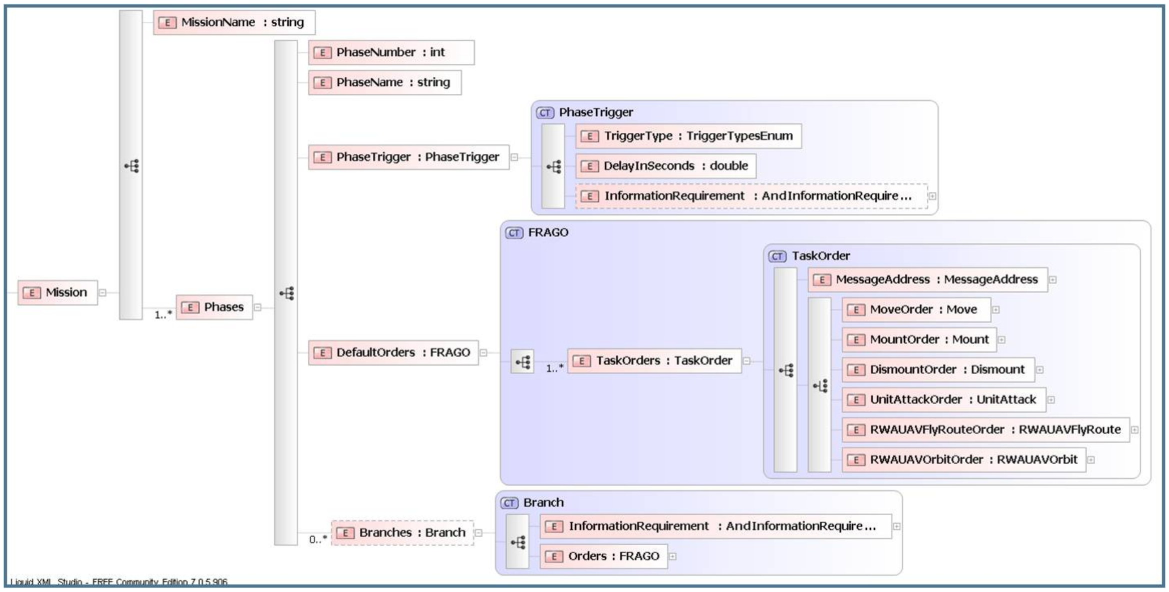

E MissionName : string
E PhaseNumber : int
E PhaseName : string
CT PhaseTrigger
E TriggerType : TriggerTypesEnum
E PhaseTrigger : PhaseTrigger
E DelayInSeconds : double
E InformationRequirement : AndInformationRequire...
CT FRAGO
CT TaskOrder
E MessageAddress : MessageAddress
E Mission
1..*
E Phases
E MoveOrder : Move
E MountOrder : Mount
E DefaultOrders : FRAGO
1..*
E TaskOrders : TaskOrder
E DismountOrder : Dismount
E UnitAttackOrder : UnitAttack
E RWAUAVFlyRouteOrder : RWAUAVFlyRoute
E RWAUAVOrbitOrder : RWAUAVOrbit
CT Branch
E InformationRequirement : AndInformationRequire...
0..*
E Branches : Branch
E Orders : FRAGO
Liquid XML Studio - FREE Community Edition 7.0.5.906

Figure 27: Mission data elements used by the command and control federate.

5 Illustrative Results

The following section shows some illustrative results that could be obtained using the PEO Soldier federation. Note that these results are illustrative and shown as a technical proof of principal - that the federation can indeed produce the data required to show an effectiveness difference resulting from different command and control systems. They were obtained using notional and unclassified data with federates that have not been subjected to verification and validation. These results should not be construed as evidence that one command and control system is better than another.

5.1 Situation Awareness Illustrative Results

A custom federate was developed to record the situation awareness picture of each leader on the battlefield. The leader of interest for this scenario is the 1st squad leader. The unmanned aircraft reconnaissance quickly identified 19 of the 23 enemy soldiers in the objective area. These positions were quickly plotted by the unmanned aircraft operator and sent over the communications network. In the GSS case, these positions were almost instantly and automatically relayed all the way down to the squad leader. In the RFI case, the manual transmission of SALUTE reports across multiple echelons made the relay of this many positions nearly impossible. Instead, friendly forces became aware of enemy forces upon contact. This occurred as the mounted forces made contact during the approach along Route Blue in the 8th to 10th minutes and again as the dismounted assault cleared the objective from the 12th to the 16 minute. Figure 28 shows the situation awareness of the 1st squad leader for each case. The gap between the red and blue lines in this graph shows opportunities for action against enemy forces. The GSS squad leader has several minutes to call for fire, plan maneuver, and move to a position of advantage.

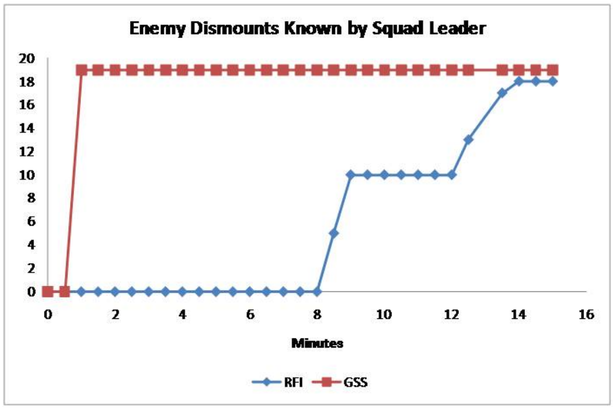


Figure 28: Comparison of enemy situation awareness for 1st squad leader for RFI and GSS cases.

5.2 Fires Illustrative Results

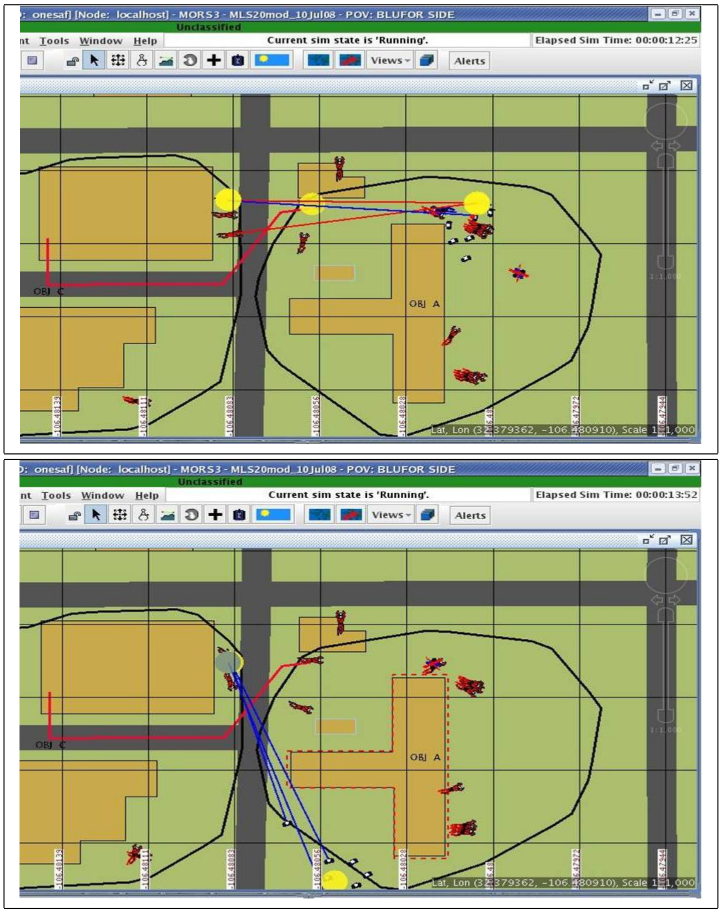
Because the GSS squad leader, becomes aware of enemy forces on the objective, several minutes before the RFI squad leader, he can initiate a call for fire much more quickly. In addition, the GSS equipped force clears and approves fires much more quickly than the RFI force. For this reason, the GSS equipped squad is able to prepare the objective with fires prior to contact. In the RFI case, the friendly force cannot generate and process fires in time to have any effect. Friendly forces are in contact and too close to the enemy for effective fires by the time the fire mission reaches the mortars.

5.3 Maneuver Illustrative Results

In the RFI case, the dismounted force did not become aware of enemy forces in NAI 1 until they made contact with them, resulting in an ambush. In the GSS case, the squad leader becomes aware of the enemy force well before he must commit his dismounted assault along Axis North or Axis South. Based upon his maneuver plan (Figure 25), the command and control federate issues a FRAGO to 1st squad to continue the attack along Axis South. This results in a much more advantageous firefight with the enemy in NAI 1, as shown in Figure 29.

5.4 Illustrative System Effectiveness

While situation awareness and the ability to impact fire and maneuver are important factors, mission level re-



: onesafj [Node: localhost] - MORS3 - MLS20mod_10Jul08 - POV: BLUFOR SIDE

Unclassified

Current sim state is 'Running'.

Elapsed Sim Time: 00:00:12:25

Lat, Lon (32.379362, -106.480910), Scale 1:1,000

: onesafj [Node: localhost] - MORS3 - MLS20mod_10Jul08 - POV: BLUFOR SIDE

Unclassified

Current sim state is 'Running'.

Elapsed Sim Time: 00:00:13:52

Lat, Lon (32.379362, -106.480910), Scale 1:1,000

Figure 29: This figure shows alternate routes taken by forces given different levels of situation awareness. In the RFI case (top), the dismounts are not aware of the enemy located in NAI 1, and they stumble into an ambush. In the GSS case (bottom), improved awareness of enemy forces allows the squad leader to issue a FRAGO moving the dismounted assault to Axis South. This is a much more advantageous engagement into the flank of enemy forces .

sults are the real indicators of the effectiveness of a command and control system. As an example, the number of friendly survivors in 1st squad was collected for both the RFI case and the GSS case. For the RFI case, the friendly force had an average of 4.3 survivors with a standard deviation of 2.08. For the GSS case, 1st squad had an average of 7.3 survivors with a standard deviation of 4.04. For this particular experiment, the results were not statistically significant.

Most importantly, this illustrative analysis shows the capability of the PEO Soldier federation to assess the impact of the GSS on the fight. It led to improved situation awareness about the enemy force. This allowed the squad leader to call for fire and adjust maneuver in a way that was not possible with the RFI case. The federation was also able to capture effectiveness metrics such as friendly and enemy casualties during the fight.

6 Planning for Academic Year 2009-2010 Work

Based on the previous work, the road ahead through May 2010 calls for continued effort in contributing to the development of the federation so that it can be used in an analysis role to assess planned soldier command and control systems. This capability would allow PEO Soldier and Marine Corps Systems Command to assess the impacts of soldier equipment alternatives on unit outcomes.

West Point will continue to lead the modeling and analysis team to perform the following three tasks for the upcoming year. A detailed breakdown of tasks is shown in ANNEX D.

- Provide analysis support to the XM-25 counter-defilade target engagement system. This support will include analysis support for validation of the basis of issue of the XM-25. Additionally, it will include the integration of America's Army software with OneSAF so that virtual prototypes of the XM-25 can be used in an analytical, as opposed to game-only, framework. A West Point cadet capstone team will support this effort.

- Provide analysis support to evaluate the effectiveness of automated support by fire position selection in a dismounted mission. This analysis will inform PM Ground Soldier decisions about the integration of automated terrain analysis capabilities into the Land Warrior System and Ground Soldier System. A West Point cadet capstone team will support this effort.

- Continue to do federated modeling and algorithm development to better assess the impacts of soldier command and control systems on mission level performance. This includes the continued improvement and development of analysis capabilities with respect to situation awareness, fires and maneuver decisions based upon situation awareness, communications, clearance of fires, and rules of engagement.

References

AMSO. 2002 (September). *Planning Guidelines for Simulation and Modeling for Acquisition, Requirements, and Training, Change 1.* Army Modeling and Simulation Office.

Boylan, Gregory L. 2006 (June). *PEO Soldier Simulation Roadmap: Continued Efforts in Implementation.* Tech. rept. DTIC ADA448073. United States Military Academy Operations Research Center, West Point, NY.

Carothers, Christopher, Fujimoto, Richard, Weatherly, Richard, & Wilson, Annette. 1997. Design and Implementation of HLA Time Management in RTI Version F.0. *In:* Andradóttir, S., Healy, K. J., Withers, D. H., & Nelson, B. L. (eds), *Proceedings of the 1997 Winter Simulation Conference.*

Copeland, Douglas. 2007a. Land Warrior DOTMLPF and LUT Results. *Infantry Magazine*, May-June, 23–29.

Copeland, Douglas. 2007b. Stryker Unit Deploys with Land Warrior. *Infantry Magazine*, May-June, 16–22.

Fujimoto, R. M. 1998. Time Management in the High Level Architecture. *Simulation*, **71**(6), 388–400.

Gallant, Scott, Metevier, Christopher, & Snively, Keith. 2009. Systems engineering for distributed simulation within MATREX. *In: Proceedings of the 2009 Spring Simulation Multiconference.* Society for Modeling and Simulation International.

HQDA. 1991 (July). *Field Manual 6-30: Techniques, Tactics, and Procedures for Observed Fire.* Headquarters, Department of the Army.

HQDA. 2002 (October). *FM 3-09.31: Tactics, Techniques, and Procedures for Fire Support for the Combined Arms Commander.* Headquarters, Department of the Army.

Hurt, Tom, McKelvey, Tim, & McDonnell, Joe. 2006. The Modeling Architecture for Technology, Research, and Experimentation. *Pages 1261–1265 of:* Perrone, L. F., Wieland, F. P., Liu, J., Lawson, B. G., Nicol, D. M., & Fujimoto, R. M. (eds), *Proceedings of the 2006 Winter Simulation Conference.*

IEEE-SA Standards Board. 1998. *Standard for Distributed Interactive Simulation - Application protocols.* Tech. rept. IEEE 1278.1A-1998.

IEEE-SA Standards Board. 2000 (September). *IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules.* Tech. rept. IEEE 1516-2000.

Institute for Electrical and Electronics Engineers. 1990. *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries.* Tech. rept. New York.

Kewley, Robert, & Tolk, Andreas. 2009. A Systems Engineering Process for Development of Federated Simulations. *In: Proceedings of the 2009 Military Modeling and Simulation Conference.*

Kewley, Robert H., Goerger, Niki, Teague, Edward, Henderson, Dale, & Cook, James. 2008. Federated Simulations for Systems of Systems Integration. *In: Proceedings of the 2008 Winter Simulation Conference.*

Kramlich, Gary. 2007. *PEO Soldier Simulation Roadmap III: Initial Working Federation.* Tech. rept. DSE-TR-0704. United States Military Academy Operations Research Center, West Point, NY.

Law, Averill M. 2007. *Simulation, Modeling, and Analysis.* 4th edn. New York: McGraw-Hill.

Martin, Grant. 2005 (July). *PEO Soldier Simulation Roadmap: Initial Steps in Implemetation.* Tech. rept. DTIC ADA435707. United States Military Academy Operations Research Center, West Point, NY.

MATREX. 2008 (November). *MATREX Federational Object Model Version 4.1 Documentation.* Modeling Architecture for Technology, Research, and Experimentation.

Object Management Group. 2007. *OMG Model Driven Architecture. http://www.omg.org/mda. Accessed 14 March 2008.*

Odersky, Martin, Altherr, Philippe, Cremet, Vincent, Dragos, Iulian, Dubochet, Gilles, Emir, Burak, McDirmid, Sean, Micheloud, Stéphane, Mihaylov, Nikolay, Schinz, Michel, Stenman, Erik, Spoon, Lex, & Zenger, Matthias. 2006. *An Overview of the Scala Programming Language, Second Edition*. Tech. rept. École Polytechnique Fédérale de Lausanne.

Parnell, Gregory, Driscoll, Patrick, & Henderson, Dale (eds). 2008. *Decision Making in Systems Engineering and Management*. Hoboken: Wiley.

Pinnell, Daniel A., & Ivanoff, Kelly W. 2003. Forward Observer Battle Drills - The Time is Now. *Field Artillery Journal*, March-April.

SEDRIS. 2007. *SEDRIS Technologies. http://www.sedris.org. Accessed 14 April 2008.*

SISO. 2008. *Military Scenario Definition Language*. Tech. rept. Simulation Interoperability Standards Organization.

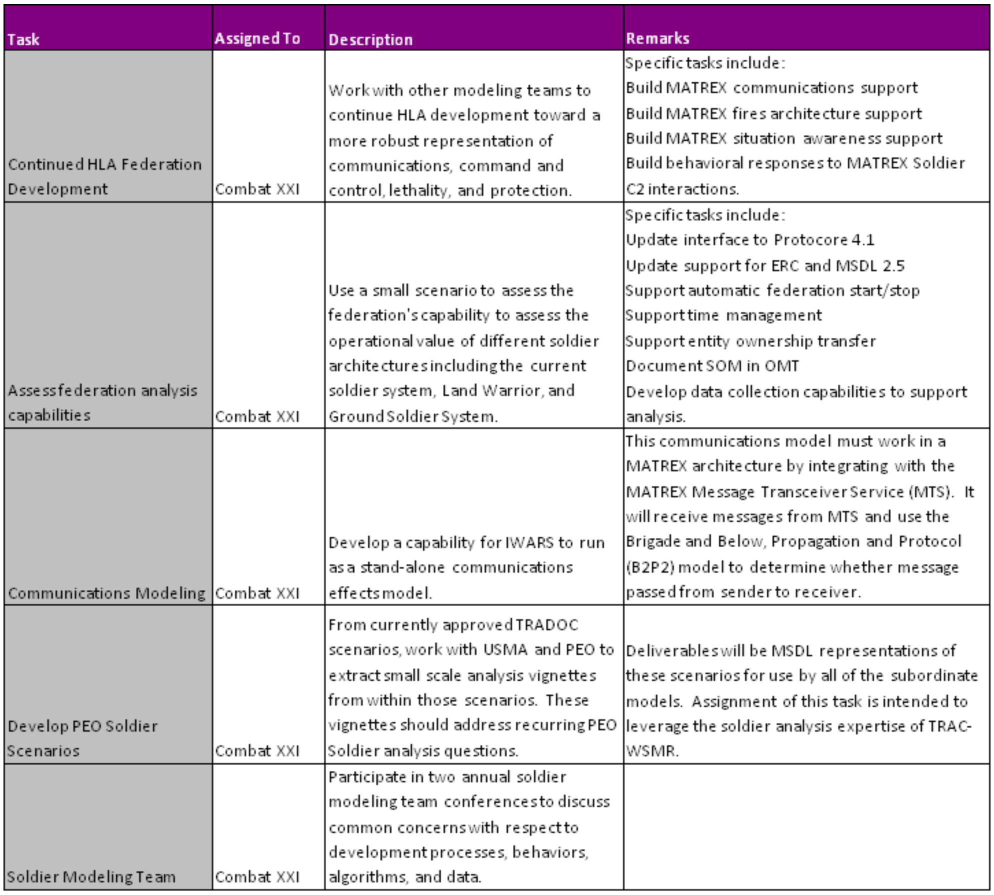
Taylor, Simon, Sharpe, Jon, & Ladbrook, John. 2003. Time Management Issues in COTS Distributed Simulation: A Case Study. *In:* Chick, S., Sanchez, P.J., Ferrin, D., & Morris, D.J. (eds), *Proceedings of the 2003 Winter Simulation Conference*.

Tollefson, Eric S., & Boylan, Greg L. 2004 (September). *Simulation Roadmap for Program Simulation Roadmap for Program Executive Office (PEO) Soldier*. Tech. rept. DTIC ADA425648. United States Military Academy Operations Research Center, West Point, NY.

Nomenclature

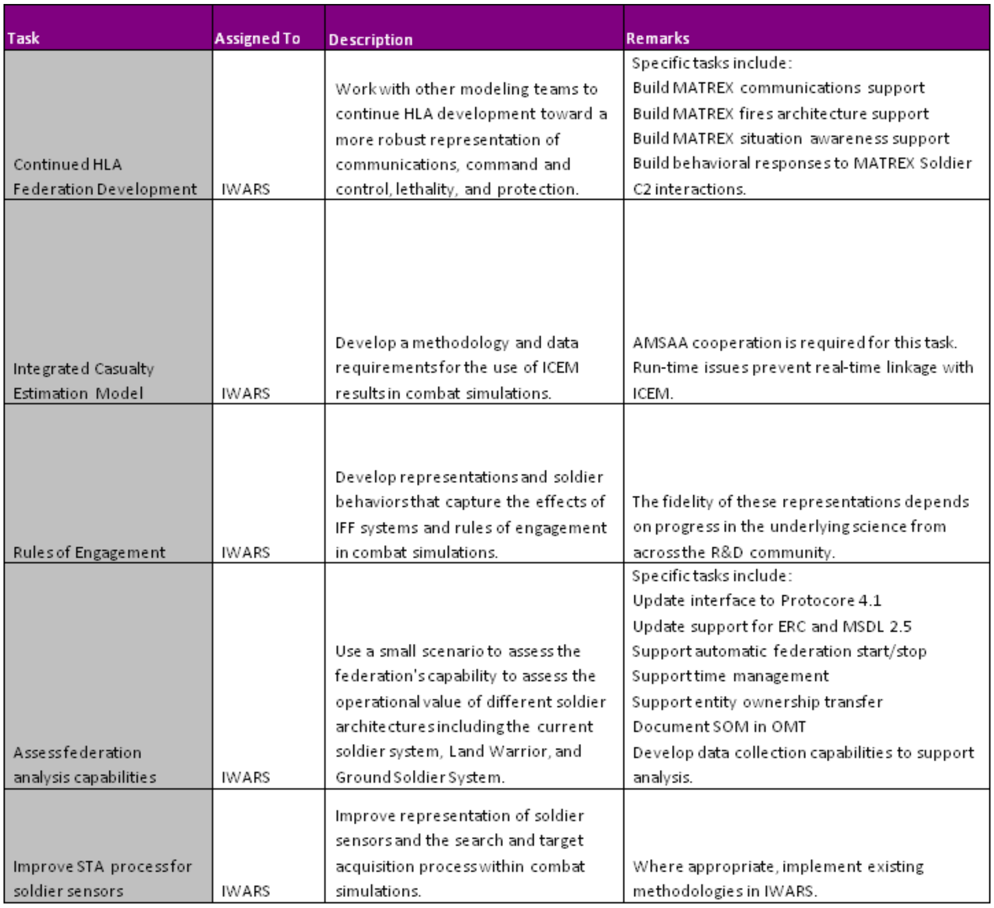
AMSAA	Army Materiel Systems Analysis Activity
ARL-SLAD	Army Research Lab - Survivability/Lethality Analysis Directorate
B2P2	Brigade and Below Propagation and Protocols
BFV	Bradley Fighting Vehicle
CERDEC	Communications-Electronics Research, Development, and Engineering Center
COMBAT^{XXI}	Combined-Arms Analysis Tool for the 21st Century
COMPOSER	Communications Planner for Operational and Simulation Effects with Realism
DDM	Data Distribution Management
ERC	Environmental Runtime Component
FDC	Fire Direction Center
FRAGO	Fragmentary Order
FSO	Fire Support Officer
GSS	Ground Soldier System
HLA	High Level Architecture
IFF	Identification Friend or Foe
IWARS	Infantry Warrior Simulation
LOP	Local Operational Picture
MATREX	Modeling Architecture for Technology Research and Experimentation
MDA	Model Driven Architectures
NAI	Named Area of Interest
OBJ	Objective
OneSAF	One Semi-Automated Forces
ORCEN	Operations Research Center of Excellence
PEO	Program Executive Office
RDECOM	Research Development and Engineering Command
RFI	Rapid Fielding Initiative
SAF	Semi-Automated Forces
SBF	Support by Fire
SMART	Simulation and Modeling for Acquisition Requirements and Training
STMS	Soldier Tactical Mission System
TRADOC	Training and Doctrine Command
UML	Unified Modeling Language

ANNEX A - Development Tasks for Academic Year 2008-2009



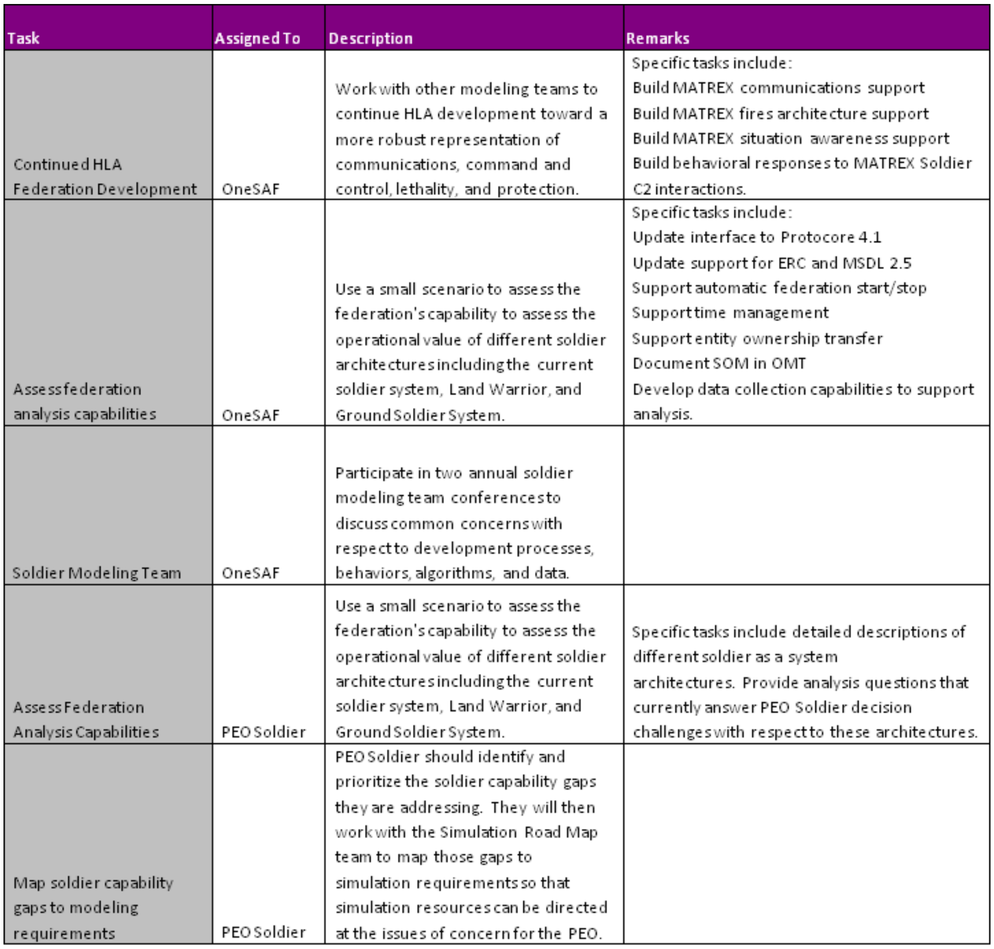
Task	Assigned To	Description	Remarks
Continued HLA Federation Development	Combat XXI	Work with other modeling teams to continue HLA development toward a more robust representation of communications, command and control, lethality, and protection.	Specific tasks include: Build MATREX communications support Build MATREX fires architecture support Build MATREX situation awareness support Build behavioral responses to MATREX Soldier C2 interactions.
Assess federation analysis capabilities	Combat XXI	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific tasks include: Update interface to Protocore 4.1 Update support for ERC and MSDL 2.5 Support automatic federation start/stop Support time management Support entity ownership transfer Document SOM in OMT Develop data collection capabilities to support analysis.
Communications Modeling	Combat XXI	Develop a capability for IWARS to run as a stand-alone communications effects model.	This communications model must work in a MATREX architecture by integrating with the MATREX Message Transceiver Service (MTS). It will receive messages from MTS and use the Brigade and Below, Propagation and Protocol (B2P2) model to determine whether message passed from sender to receiver.
Develop PEO Soldier Scenarios	Combat XXI	From currently approved TRADOC scenarios, work with USMA and PEO to extract small scale analysis vignettes from within those scenarios. These vignettes should address recurring PEO Soldier analysis questions.	Deliverables will be MSDL representations of these scenarios for use by all of the subordinate models. Assignment of this task is intended to leverage the soldier analysis expertise of TRAC-WSMR.
Soldier Modeling Team	Combat XXI	Participate in two annual soldier modeling team conferences to discuss common concerns with respect to development processes, behaviors, algorithms, and data.	

Figure 30: COMBAT^{XXI} tasks for academic year 2008-2009.



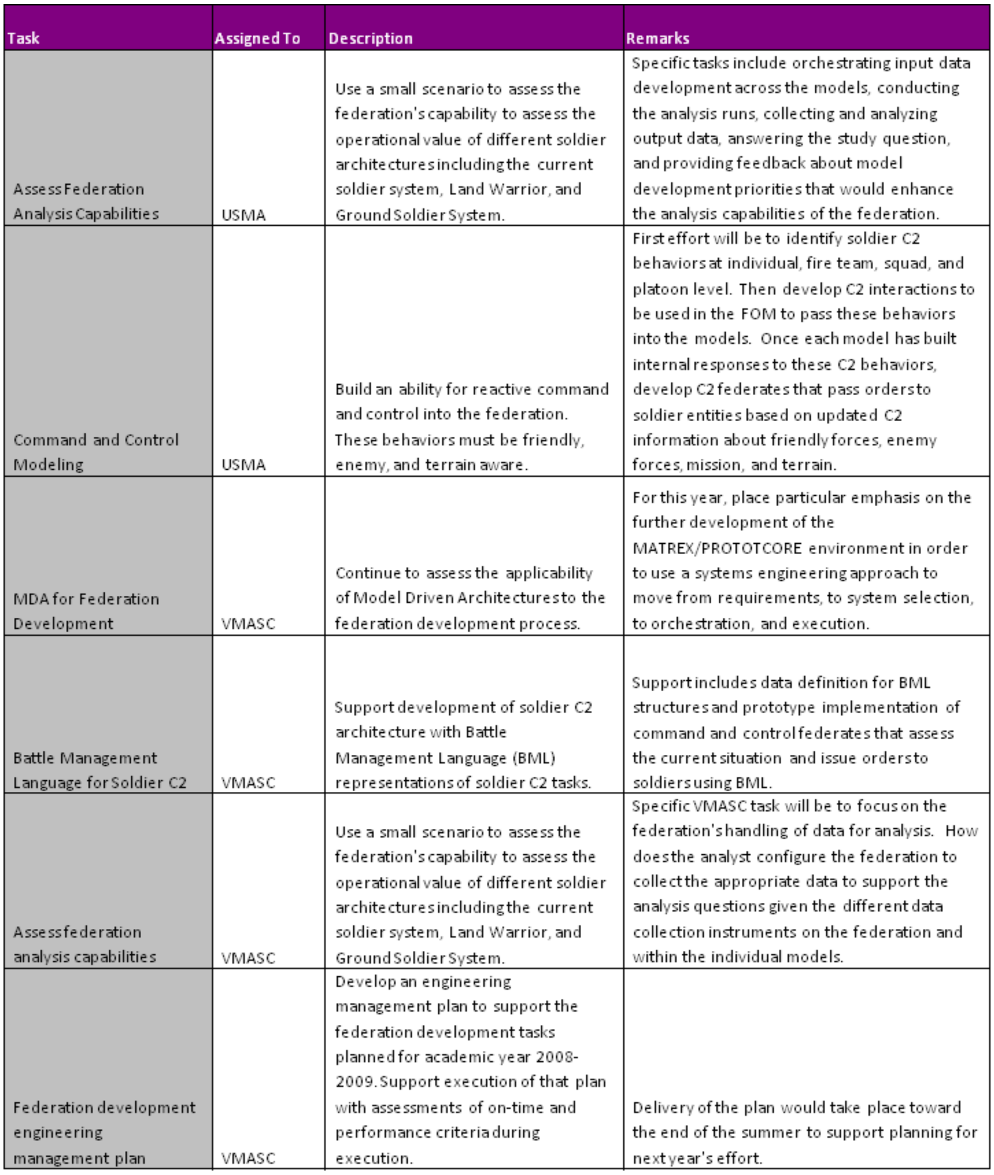
Task	Assigned To	Description	Remarks
Continued HLA Federation Development	IWARS	Work with other modeling teams to continue HLA development toward a more robust representation of communications, command and control, lethality, and protection.	Specific tasks include: Build MATREX communications support Build MATREX fires architecture support Build MATREX situation awareness support Build behavioral responses to MATREX Soldier C2 interactions.
Integrated Casualty Estimation Model	IWARS	Develop a methodology and data requirements for the use of ICEM results in combat simulations.	AMSAA cooperation is required for this task. Run-time issues prevent real-time linkage with ICEM.
Rules of Engagement	IWARS	Develop representations and soldier behaviors that capture the effects of IFF systems and rules of engagement in combat simulations.	The fidelity of these representations depends on progress in the underlying science from across the R&D community.
Assess federation analysis capabilities	IWARS	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific tasks include: Update interface to Protocore 4.1 Update support for ERC and MSDL 2.5 Support automatic federation start/stop Support time management Support entity ownership transfer Document SOM in OMT Develop data collection capabilities to support analysis.
Improve STA process for soldier sensors	IWARS	Improve representation of soldier sensors and the search and target acquisition process within combat simulations.	Where appropriate, implement existing methodologies in IWARS.

Figure 31: IWARS development tasks for academic year 2008-2009



Task	Assigned To	Description	Remarks
Continued HLA Federation Development	OneSAF	Work with other modeling teams to continue HLA development toward a more robust representation of communications, command and control, lethality, and protection.	Specific tasks include: Build MATREX communications support Build MATREX fires architecture support Build MATREX situation awareness support Build behavioral responses to MATREX Soldier C2 interactions.
Assess federation analysis capabilities	OneSAF	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific tasks include: Update interface to Protocore 4.1 Update support for ERC and MSDL 2.5 Support automatic federation start/stop Support time management Support entity ownership transfer Document SOM in OMT Develop data collection capabilities to support analysis.
Soldier Modeling Team	OneSAF	Participate in two annual soldier modeling team conferences to discuss common concerns with respect to development processes, behaviors, algorithms, and data.	
Assess Federation Analysis Capabilities	PEO Soldier	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific tasks include detailed descriptions of different soldier as a system architectures. Provide analysis questions that currently answer PEO Soldier decision challenges with respect to these architectures.
Map soldier capability gaps to modeling requirements	PEO Soldier	PEO Soldier should identify and prioritize the soldier capability gaps they are addressing. They will then work with the Simulation Road Map team to map those gaps to simulation requirements so that simulation resources can be directed at the issues of concern for the PEO.	

Figure 32: OneSAF and PEO soldier tasks for academic year 2008-2009



Task	Assigned To	Description	Remarks
Assess Federation Analysis Capabilities	USMA	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific tasks include orchestrating input data development across the models, conducting the analysis runs, collecting and analyzing output data, answering the study question, and providing feedback about model development priorities that would enhance the analysis capabilities of the federation.
Command and Control Modeling	USMA	Build an ability for reactive command and control into the federation. These behaviors must be friendly, enemy, and terrain aware.	First effort will be to identify soldier C2 behaviors at individual, fire team, squad, and platoon level. Then develop C2 interactions to be used in the FOM to pass these behaviors into the models. Once each model has built internal responses to these C2 behaviors, develop C2 federates that pass orders to soldier entities based on updated C2 information about friendly forces, enemy forces, mission, and terrain.
MDA for Federation Development	VMASC	Continue to assess the applicability of Model Driven Architectures to the federation development process.	For this year, place particular emphasis on the further development of the MATREX/PROTOTCORE environment in order to use a systems engineering approach to move from requirements, to system selection, to orchestration, and execution.
Battle Management Language for Soldier C2	VMASC	Support development of soldier C2 architecture with Battle Management Language (BML) representations of soldier C2 tasks.	Support includes data definition for BML structures and prototype implementation of command and control federates that assess the current situation and issue orders to soldiers using BML.
Assess federation analysis capabilities	VMASC	Use a small scenario to assess the federation's capability to assess the operational value of different soldier architectures including the current soldier system, Land Warrior, and Ground Soldier System.	Specific VMASC task will be to focus on the federation's handling of data for analysis. How does the analyst configure the federation to collect the appropriate data to support the analysis questions given the different data collection instruments on the federation and within the individual models.
Federation development engineering management plan	VMASC	Develop an engineering management plan to support the federation development tasks planned for academic year 2008-2009. Support execution of that plan with assessments of on-time and performance criteria during execution.	Delivery of the plan would take place toward the end of the summer to support planning for next year's effort.

Figure 33: USMA and VMASC tasks for academic year 2008-2009

ANNEX B - Project Plan for Academic Year 2008-2009

ID	ℹ	Task Name	Duration	Start	Finish	Predecessors	6/1
1		**PEO Soldier Simulation Road Map**	**480 days?**	**Mon 6/16/08**	**Fri 4/16/10**		
2		**Roll AY2008 Changes into baseline releases**	**81 days?**	**Mon 6/16/08**	**Mon 10/6/08**		
3		**IWARS Modeling**	**71 days**	**Mon 6/16/08**	**Mon 9/22/08**		
4		Roll AY2008 Modeling into IWARS 1.5	1 mon	Mon 6/16/08	Fri 7/11/08		
5		Update IWARS to use Protocore 4.1 with associated FOM	2 wks	Mon 7/14/08	Fri 7/25/08	4	
6		Update IWARS to use ERC 2.5 (Version that ships with OneSAF 2.5)	2 wks	Tue 9/9/08	Mon 9/22/08	11	
7		Update IWARS to use MSDL 2.5 (Version that ships with OneSAF 2.5)	2 wks	Tue 9/9/08	Mon 9/22/08	12	
8		**OneSAF Modeling**	**70 days?**	**Mon 6/16/08**	**Fri 9/19/08**		
9		Roll AY2008 Modeling into OneSAF 2.5	3 mons	Mon 6/16/08	Fri 9/5/08		
10		Update OneSAF to use MATREX FOM 4.1	2 wks	Mon 9/8/08	Fri 9/19/08	9	
11		Release ERC 2.5	1 day?	Mon 9/8/08	Mon 9/8/08	9	
12		Release MSDL 2.5	1 day?	Mon 9/8/08	Mon 9/8/08	9	
13		**Combat XXI Modeling**	**81 days**	**Mon 6/16/08**	**Mon 10/6/08**		
14		Replicate AY2008 Modeling in Combat XXI	3 mons	Mon 6/16/08	Fri 9/5/08		
15		Update Combat XXI to use Protocore 4.1 with associated FOM	2 wks	Mon 9/8/08	Fri 9/19/08	14	
16		Update Combat XXI to use ERC 2.5 (Version that ships with OneSAF 2.5)	1 mon	Tue 9/9/08	Mon 10/6/08	11	
17		Update Combat XXI to use MSDL 2.5 (Version that ship with OneSAF 2.5)	1 mon	Tue 9/9/08	Mon 10/6/08	12	
18		**Federation Analysis and Technical Capabilities**	**50 days**	**Mon 7/28/08**	**Fri 10/3/08**		
19		**IWARS Modeling**	**10 days**	**Mon 7/28/08**	**Fri 8/8/08**		
20		IWARS Automatic Federation Start/Stop Capability	2 wks	Mon 7/28/08	Fri 8/8/08	5	
21		IWARS Time mangement to allow faster than real time run sets	2 wks	Mon 7/28/08	Fri 8/8/08	5	
22		IWARS Passing of entity ownership	2 wks	Mon 7/28/08	Fri 8/8/08	5	
23		Document IWARS internal SOM in OMT	2 wks	Mon 7/28/08	Fri 8/8/08	5	
24		**OneSAF Modeling**	**10 days**	**Mon 9/22/08**	**Fri 10/3/08**		
25		OneSAF Automatic Federation Start/Stop Capability	2 wks	Mon 9/22/08	Fri 10/3/08	10	
26		OneSAF Time mangement to allow faster than real time run sets	2 wks	Mon 9/22/08	Fri 10/3/08	10	
27		OneSAF Passing of entity ownership	2 wks	Mon 9/22/08	Fri 10/3/08	10	
28		Document OneSAF internal SOM in OMT	2 wks	Mon 9/22/08	Fri 10/3/08	10	
29		**Combat XXI Modeling**	**10 days**	**Mon 9/22/08**	**Fri 10/3/08**		
30		CombatXXI Automatic Federation Start/Stop Capability	2 wks	Mon 9/22/08	Fri 10/3/08	15	
31		CombatXXI Time mangement to allow faster than real time run sets	2 wks	Mon 9/22/08	Fri 10/3/08	15	
32		CombatXXI Passing of entity ownership	2 wks	Mon 9/22/08	Fri 10/3/08	15	

Project: Simulation Road Map 2009
Date: Wed 6/11/08

Task	▭	Milestone	◆	External Tasks	▭
Split	┈	Summary	▬	External Milestone	◆
Progress	▬	Project Summary	▬	Deadline	⇩

ID	ⓘ	Task Name	Duration	Start	Finish	Predecessors	6/1
33		Document CombatXXI internal SOM in OMT	2 wks	Mon 9/22/08	Fri 10/3/08	15	
34		Federation Test Event 1 (Test AY2008 scenario updated to Protocore 4.1 and latest models)	1 wk	Mon 10/6/08	Fri 10/10/08	29,24,19	
35		**Systems Engineering**	**115 days**	**Mon 6/16/08**	**Fri 11/21/08**		
36		Identify soldier systems under consideration for FY10 and FY11	1 mon	Mon 6/16/08	Fri 7/11/08		
37		Identify soldier missions and tasks effected by soldier systems under consideration	1 mon	Mon 7/14/08	Fri 8/8/08	36	
38		Develop Analysis Scenario	1 mon	Mon 8/11/08	Fri 9/5/08	37	
39		Identify how to measure capabilities increases and decreases in analysis scenario	1 wk	Mon 9/8/08	Fri 9/12/08	38	
40		Capture missions and means in analysis scenario in UML	2 wks	Mon 9/15/08	Fri 9/26/08	39	
41		Allocate functional capbilities to federates	2 wks	Mon 9/29/08	Fri 10/10/08	40	
42		Design federation in Protocore tools to support requirements	1 mon	Mon 10/13/08	Fri 11/7/08	41	
43		Generate code stubs to enable federate integration	1 wk	Mon 11/10/08	Fri 11/14/08	42	
44		Generate test cases to enable federation integration	1 wk	Mon 11/17/08	Fri 11/21/08	43	
45		**Command and Control Modeling**	**200 days**	**Mon 6/16/08**	**Fri 3/20/09**		
46		**Soldier Communications Modeling**	**125 days**	**Mon 6/16/08**	**Fri 12/5/08**		
47		Gain understanding of MTS and its use in MATREX	1 mon	Mon 6/16/08	Fri 7/11/08		
48		Identify appropriate communications effects model analysis of soldier communications	2 mons	Mon 7/14/08	Fri 9/5/08	47	
49		Integrate commincations model with MTS	2 mons	Mon 9/8/08	Fri 10/31/08	48	
50		Build communications test case	1 wk	Mon 11/3/08	Fri 11/7/08	49	
51		**Build MATREX communications support into federates**	**20 days**	**Mon 11/3/08**	**Fri 11/28/08**	**34,49**	
52		Build MATREX communictions support into IWARS	1 mon	Mon 11/3/08	Fri 11/28/08		
53		Build MATREX communictions support into OneSAF	1 mon	Mon 11/3/08	Fri 11/28/08		
54		Build MATREX communictions support into CombatXXI	1 mon	Mon 11/3/08	Fri 11/28/08		
55		Test communicatons functionality using communications test scenario	1 wk	Mon 12/1/08	Fri 12/5/08	52,53,54,49,50	
56		**Modeling of the Fires Process**	**75 days**	**Mon 8/11/08**	**Fri 11/21/08**		
57		Identify functional tasks supporting fires process and capture in UML	1 mon	Mon 8/11/08	Fri 9/5/08	37	
58		Allocate fires functional capabilities to federates	1 mon	Mon 9/8/08	Fri 10/3/08	57	
59		Build fires test scenario	2 wks	Mon 10/6/08	Fri 10/17/08	58	
60		**Build MATREX fires support into federates**	**20 days**	**Mon 10/20/08**	**Fri 11/14/08**	**59,34**	
61		Build MATREX fires support into IWARS	1 mon	Mon 10/20/08	Fri 11/14/08		
62		Build MATREX fires support into OneSAF	1 mon	Mon 10/20/08	Fri 11/14/08		
63		Build MATREX fires support into CombatXXI	1 mon	Mon 10/20/08	Fri 11/14/08		
64		Test fires functionality using firest test scenario	1 wk	Mon 11/17/08	Fri 11/21/08	60	

Project: Simulation Road Map 2009
Date: Wed 6/11/08

Task	▭	Milestone	◆	External Tasks	▭
Split	┈	Summary	▬	External Milestone	◆
Progress	▬	Project Summary	▬	Deadline	⇩

ID	ℹ	Task Name	Duration	Start	Finish	Predecessors	6/1
65		**Situation Awareness Modeling**	**115 days**	**Mon 6/16/08**	**Fri 11/21/08**		
66		Gain an understanding of C3 Grid and its use in MATREX	2 mons	Mon 6/16/08	Fri 8/8/08		
67		Identify functional tasks supporting situation awareness and capture in UML	1 mon	Mon 8/11/08	Fri 9/5/08	37	
68		Allocate situation awareness functional capabilities to federates	1 mon	Mon 9/8/08	Fri 10/3/08	67	
69		Build SA test scenario	2 wks	Mon 10/6/08	Fri 10/17/08	68	
70		**Build MATREX SA support into federates**	**20 days**	**Mon 10/20/08**	**Fri 11/14/08**	**69,34**	
71		Build MATREX SA support into IWARS	1 mon	Mon 10/20/08	Fri 11/14/08		
72		Build MATREX SA support into OneSAF	1 mon	Mon 10/20/08	Fri 11/14/08		
73		Build MATREX SA support into CombatXXI	1 mon	Mon 10/20/08	Fri 11/14/08		
74		Capture and model soldier SA in C3 Grid	1 mon	Mon 10/20/08	Fri 11/14/08		
75		Test SA functionality using firest test scenario	1 wk	Mon 11/17/08	Fri 11/21/08	70	
76		Federation test event 2 (Show communications, fires, and SA functionality using AY2008 sce	1 wk	Mon 12/8/08	Fri 12/12/08	46,56,65	
77		**Command and Control Decision Modeling**	**120 days**	**Mon 9/29/08**	**Fri 3/13/09**		
78		Identify functional tasks supporting C2 and capture in UML	1 mon	Mon 9/29/08	Fri 10/24/08	40	
79		Allocate C2 functional capabilities to federates	1 mon	Mon 10/27/08	Fri 11/21/08	78	
80		Build C2 data structures for inclusion in MATREX FOM	1 mon	Mon 11/24/08	Fri 12/19/08	79	
81		Extend MATREX FOM to support soldier C2 data structures	1 wk	Mon 12/22/08	Fri 12/26/08	80	
82		Build C2 test scenario	2 wks	Mon 12/29/08	Fri 1/9/09	81	
83		**Build MATREX C2 support into federates**	**40 days**	**Mon 1/12/09**	**Fri 3/6/09**	**82**	
84		Build MATREX C2 support into IWARS	1 mon	Mon 1/12/09	Fri 2/6/09		
85		Build MATREX C2 support into OneSAF	1 mon	Mon 1/12/09	Fri 2/6/09		
86		Build MATREX C2 support into CombatXXI	1 mon	Mon 1/12/09	Fri 2/6/09		
87		Develop C2 federate to make appropriate C2 decisions	2 mons	Mon 1/12/09	Fri 3/6/09		
88		Test C2 functionality using firest test scenario	1 wk	Mon 3/9/09	Fri 3/13/09	83	
89		Federation test event 2 (Show C2 functionality using th AY2009 scenario)	1 wk	Mon 3/16/09	Fri 3/20/09	77	
90		ICEM Casualty Modeling Capability	8 mons	Mon 6/16/08	Fri 1/23/09		
91		IFF and Rules of Engagement Modeling	8 mons	Mon 6/16/08	Fri 1/23/09		
92		Improve STA Process for Soldier Sensors	8 mons	Mon 6/16/08	Fri 1/23/09		
93		Model Soldier Power Consumption	8 mons	Mon 6/16/08	Fri 1/23/09		
94		Build federation data collection capability	8 mons	Mon 6/16/08	Fri 1/23/09		
95		Conduct analysis using the developed federation	2 mons	Mon 3/23/09	Fri 5/15/09	91,92,93,94,89,76	
96		Verify and validate the federation	12 mons	Mon 5/18/09	Fri 4/16/10	95	

Project: Simulation Road Map 2009
Date: Wed 6/11/08

Task	▭	Milestone	◆	External Tasks	▬
Split	┈	Summary	▬	External Milestone	◆
Progress	▬	Project Summary	▬	Deadline	⇩

ANNEX C - Sample MATREX Test Case for Mount/Dismount Function

Author	Test Case Category
ATC Generated Tue Mar 31 15:21:57 EDT 2009	ATC

Purpose/Objective

(Description of the purpose/objective for this test case)

This test case tests IWARS capability to mount and entity on a OneSAF vehicle. This test can be executed using the entitties as defined by the MSDL file WhiteSandsAttack3.xml. Specifically, the dismounted soldiers "B/2/1/SAW" and "B/2/1/GRND" will mount the vehcle "Stryker PL WM" .

The chain of events will proceed as follows.

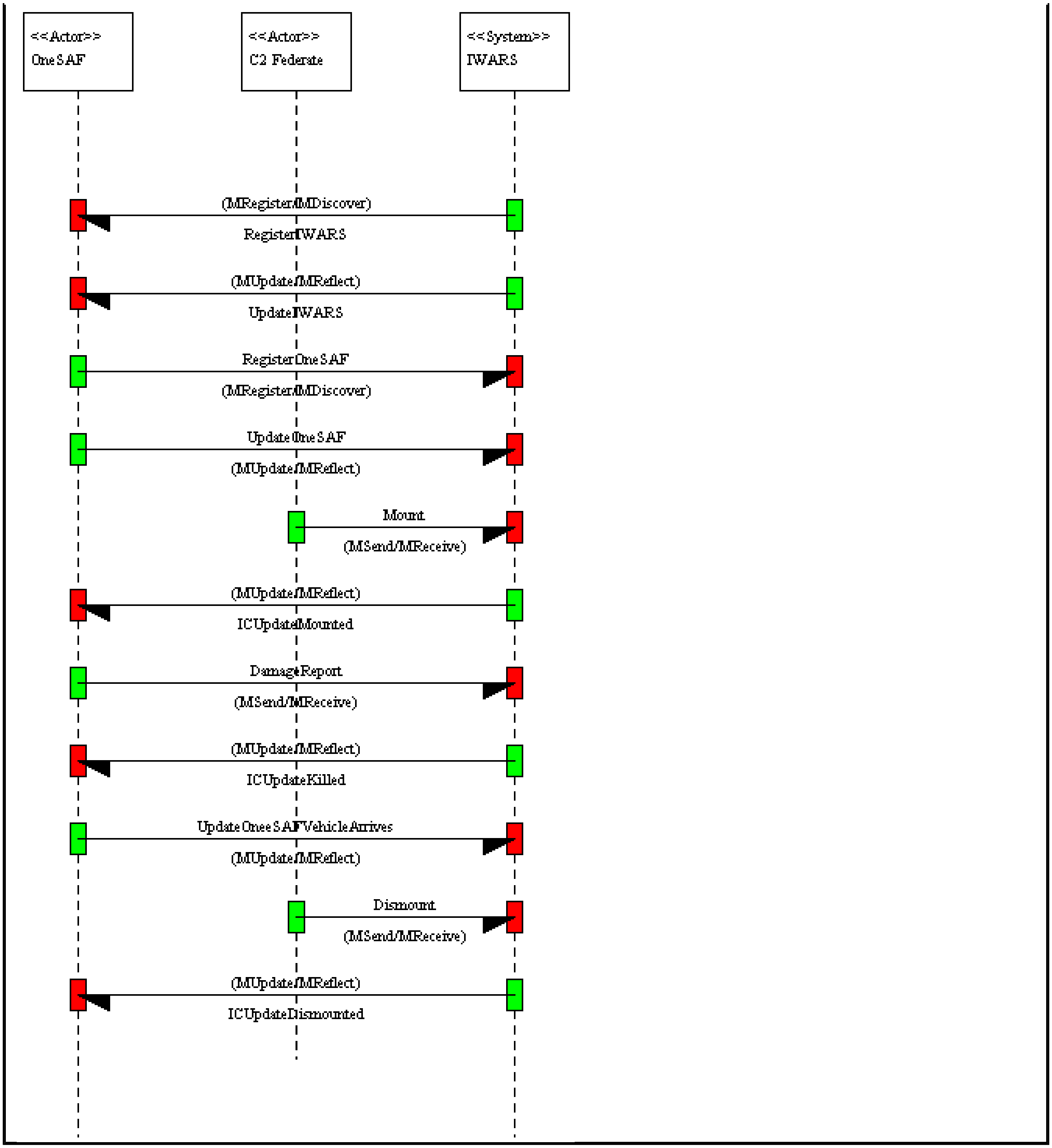
1. C2 Federate issues a Mount interaction with MountDismountCommand set to "Mount"
2. IWARS moves soldiers to mount point.
3. When soldier is within 10 meters of the vehicle, IWARS mounts the soldier.
4. The soldier cannot perform combat functions inside the vehicle, so IWARS disables the soldier for target acquisition, firing, damage (which will be computed by OneSAF), and display. However, IWARS continues to update the StateVector of these entities to match the StateVector of the vehilce they ride. IWARS continues to publish IndividualCombatant updates based on this StateVector.
5. IWARS sends a IndividualCombatantUpdate to reflect mounted status of the soldier on the designated vehicle.
6. The vehicle begins to move.
7. The vehicle is hit by an RPG, causing OneSAF to assess one of the mounted soldiers as a K_Kill.
8. OneSAF sends a DamageReport for the mounted soldier.
9. IWARS updates the damage status of the entity based on the DamageReport.
10. The C2 Federate issues a Dismount command.
11. The vehicle stops.
12. IWARS soldier dismounts the vehicle.
13. IWARS reactivates the soldier for target acquisition, firing, damage, and display.
14. IWARS sends an IndividualCombatantUpdate to relfect the dismunted status of the soldier.

System/Actor Notes:

Federate Name	Notes
IWARS	
OneSAF*	
C2 Federate*	

* indicates played by ATC

Sequence Diagram



Entrance Criteria (Conditions/products required in order to support the execution of this test)	**Successful** **(√)**
IWARS ready and available RTI software ready and available ATC test case ready and available	[ ]

#	**Setup Steps** (Steps needed to setup the conditions of the test)	**Successful**

(√)

1

Notes:
This checks to be sure IWARS is registering the entities B/2/1/SAW and B/2/1/GRND

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
RegisterIWARS	IWARS	OneSAF	0	0	60	-

Parameter Name	Validated Value
PlatformID	B/2/1/SAW

[ ]

2

Notes:
This checks to be sure IWARS is registering the entities B/2/1/SAW and B/2/1/GRND

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
RegisterIWARS	IWARS	OneSAF	0	0	60	-

Parameter Name	Validated Value
PlatformID	B/2/1/GRND

[ ]

3

Notes:
This interaction ensures IWARS is sending IndividualCombatant updates for the entities "B/2/1/SAW" and "B/2/1/GRND

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
UpdateIWARS	IWARS	OneSAF	0	0	20	RegisterIWARS RegisterIWARS

Parameter Name	Validated Value
PlatformID	B/2/1/SAW

[ ]

4

Notes:
This interaction ensures IWARS is sending IndividualCombatant updates for the entities "B/2/1/SAW" and "B/2/1/GRND

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
UpdateIWARS	IWARS	OneSAF	0	0	20	RegisterIWARS RegisterIWARS

Parameter Name	Validated Value
PlatformID	B/2/1/GRND

[ ]

5

Notes:
Registers the vehicle "Stryker PL WM" for discovery by IWARS.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
RegisterOneSAF	OneSAF	IWARS	5	0	0	UpdateIWARS UpdateIWARS

[ ]

6

Notes:

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
UpdateOneSAF	OneSAF	IWARS	5	0	0	RegisterOneSAF

[ ]

7

Notes:
Mount command from Squad Leader, 2/1/SQDLDR to B/2/1/SAW and B/2/1/GRND to mount the vehicle "Stryker PL WM". This should initiate movement of both entities to the mount point. When within 10m of the vehcicle, the entities should be mounted.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
Mount	C2 Federate	IWARS	15	0	0	UpdateOneSAF

[ ]

8

Notes:
Once mounted, IWARS sends an IndividualCombatant update for each entity with Mounted=TRUE and MountedOn="Stryker PL WM". At this time, IWARS should also disable each entity for target acquisition, firing, damage, and display. However, IWARS continues to update the StateVector of these entities to match the StateVector of the vehilce they ride. IWARS continues to publish IndividualCombatant updates based on this StateVector.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
ICUpdateMounted	IWARS	OneSAF	0	0	300	Mount

Parameter Name	Validated Value
Mounted	true
MountedOn	Stryker PL WM
PlatformID	B/2/1/SAW

[ ]

9

Notes:
Once mounted, IWARS sends an IndividualCombatant update for each entity with Mounted=TRUE and MountedOn="Stryker PL WM". At this time, IWARS should also disable each entity for target acquisition, firing, damage, and display. However, IWARS continues to update the StateVector of these entities to match the StateVector of the vehilce they ride. IWARS continues to publish IndividualCombatant updates based on this StateVector.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
ICUpdateMounted	IWARS	OneSAF	0	0	300	Mount

Parameter Name	Validated Value
Mounted	true
MountedOn	Stryker PL WM
PlatformID	B/2/1/GRND

[ ]

10

Notes:
Based on an RPG hit, OneSAF comutes damage to one of the mounted entities and passes the report so that IWARS can update its status.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
DamageReport	OneSAF	IWARS	5	0	0	ICUpdateMounted ICUpdateMounted

[ ]

11

Notes:
Once the damage interaction is received by IWARS, IWARS updates the status of B/2/1/SAW to K-Kill

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
ICUpdateKilled	IWARS	OneSAF	0	0	30	DamageReport

Parameter Name	Validated Value
DamageState	K_Kill
PlatformID	B/2/1/SAW

[ ]

12

Notes:
OneSAF vehcile updates its position to arrive at the dismount point.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
UpdateOneeSAFVehicleArrives	OneSAF	IWARS	20	0	0	ICUpdateKilled

[ ]

13

Notes:
C2 Federate issues dismount command to dismount entities from Stryker PL WM. Only B/2/1/GRND should dismount, as B/2/1/SAW has been killed.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
Dismount	C2 Federate	IWARS	5	0	0	UpdateOneeSAFVehicleArrives

[ ]

14

Notes:
Once the dismount command is executed, IWARS updates the status of B/2/1/GRND to Mounted=FALSE. Also, IWARS should enable this entity once again for target acquisition, firing, damage, and display.

ATC Event:

Name	Send	Receive	Delay	MinWait	MaxWait	Dependencies
ICUpdateDismounted	IWARS	OneSAF	0	0	60	Dismount

Parameter Name	Validated Value
Mounted	false
PlatformID	B/2/1/GRND

[ ]

Execution Steps

#	Step needed to execute the test	Completed (√)	Expected result for this step	Successful (√)
1	Start RTI executive	[ ]	RTI executive and RTI console up and running	[ ]
2	Start IWARS	[ ]	Federate joins the current Federation	[ ]
3	Start ATC test federate	[ ]	Federate joins the current Federation	[ ]
4	Watch ATC output for test results	[ ]	ATC Test indicates PASSED.	[ ]

Expected Results

(Summary description of the expected outcome for this test case)

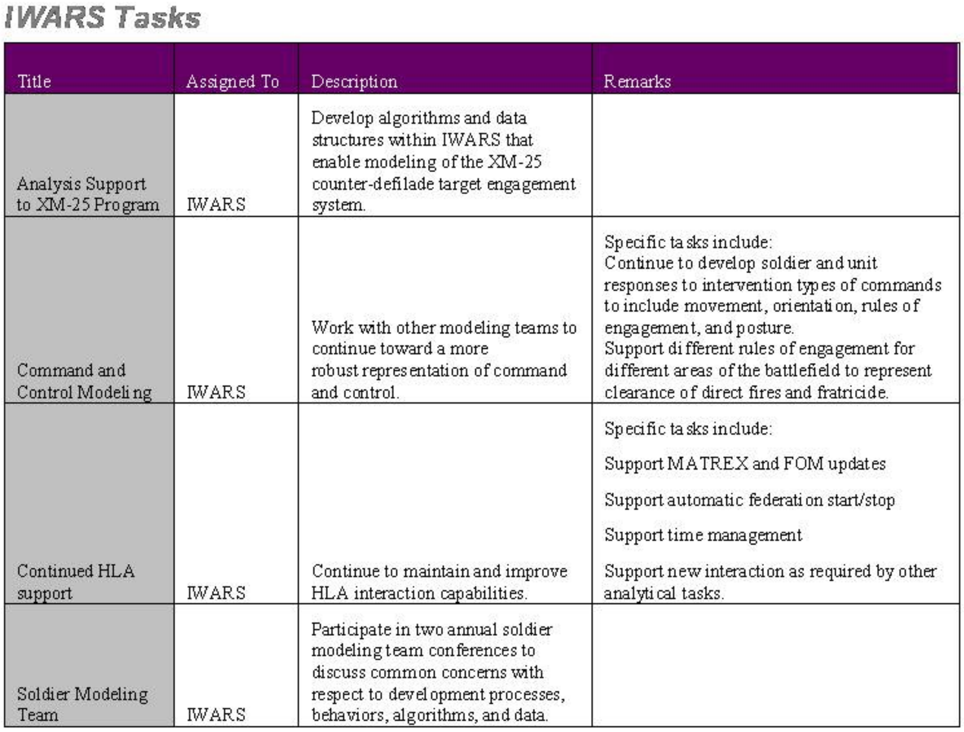
ATC Test Passed!

ACTUAL RESULTS

(Document results as needed in addition to the ï¿½checkedï¿½ items above)

TEST EXECUTION RECORD (Sign-off of tester and test witness)			
Date:		Tester	Witness
Printed Name:			
Signature:			

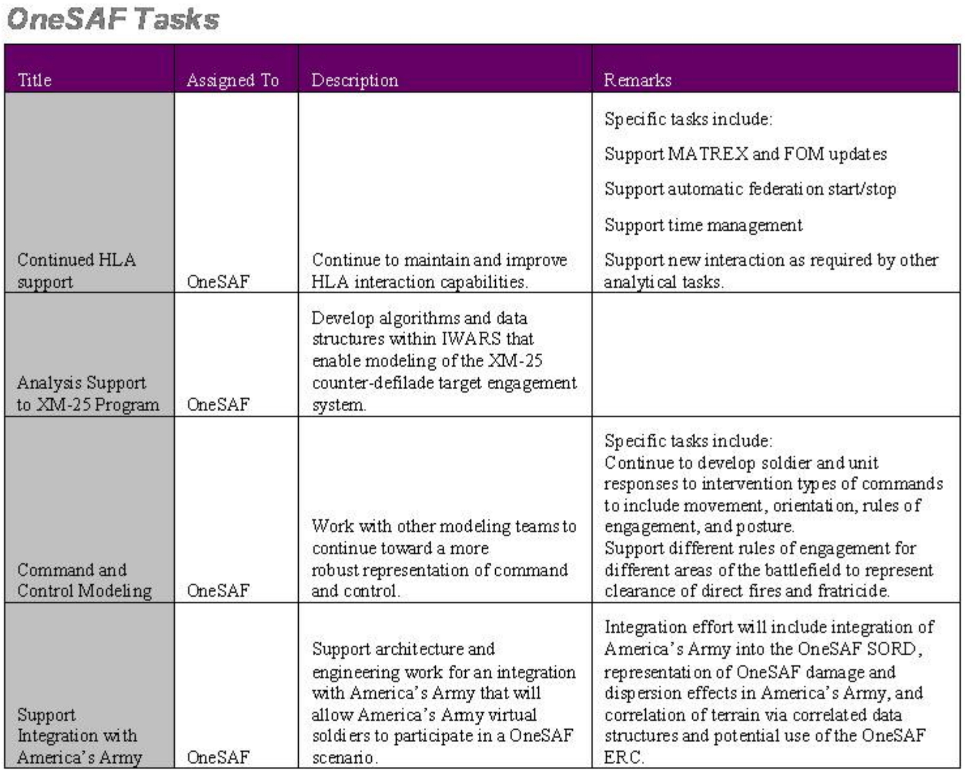
ANNEX D - Planned Tasks for Academic Year 2009-2010



IWARS Tasks

Title	Assigned To	Description	Remarks
Analysis Support to XM-25 Program	IWARS	Develop algorithms and data structures within IWARS that enable modeling of the XM-25 counter-defilade target engagement system.	
Command and Control Modeling	IWARS	Work with other modeling teams to continue toward a more robust representation of command and control.	Specific tasks include: Continue to develop soldier and unit responses to intervention types of commands to include movement, orientation, rules of engagement, and posture. Support different rules of engagement for different areas of the battlefield to represent clearance of direct fires and fratricide.
Continued HLA support	IWARS	Continue to maintain and improve HLA interaction capabilities.	Specific tasks include: Support MATREX and FOM updates Support automatic federation start/stop Support time management Support new interaction as required by other analytical tasks.
Soldier Modeling Team	IWARS	Participate in two annual soldier modeling team conferences to discuss common concerns with respect to development processes, behaviors, algorithms, and data.	

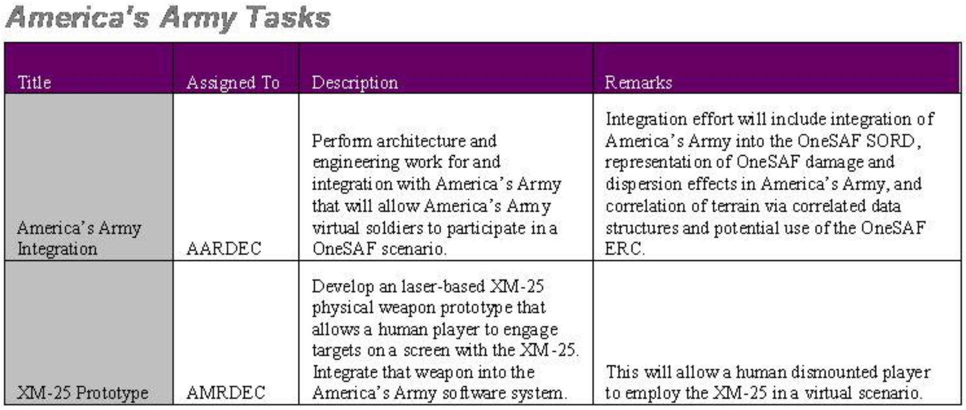
Figure 34: IWARS tasks for academic year 2009-2010.



OneSAF Tasks

Title	Assigned To	Description	Remarks
Continued HLA support	OneSAF	Continue to maintain and improve HLA interaction capabilities.	Specific tasks include: Support MATREX and FOM updates Support automatic federation start/stop Support time management Support new interaction as required by other analytical tasks.
Analysis Support to XM-25 Program	OneSAF	Develop algorithms and data structures within IWARS that enable modeling of the XM-25 counter-defilade target engagement system.	
Command and Control Modeling	OneSAF	Work with other modeling teams to continue toward a more robust representation of command and control.	Specific tasks include: Continue to develop soldier and unit responses to intervention types of commands to include movement, orientation, rules of engagement, and posture. Support different rules of engagement for different areas of the battlefield to represent clearance of direct fires and fratricide.
Support Integration with America's Army	OneSAF	Support architecture and engineering work for an integration with America's Army that will allow America's Army virtual soldiers to participate in a OneSAF scenario.	Integration effort will include integration of America's Army into the OneSAF SORD, representation of OneSAF damage and dispersion effects in America's Army, and correlation of terrain via correlated data structures and potential use of the OneSAF ERC.

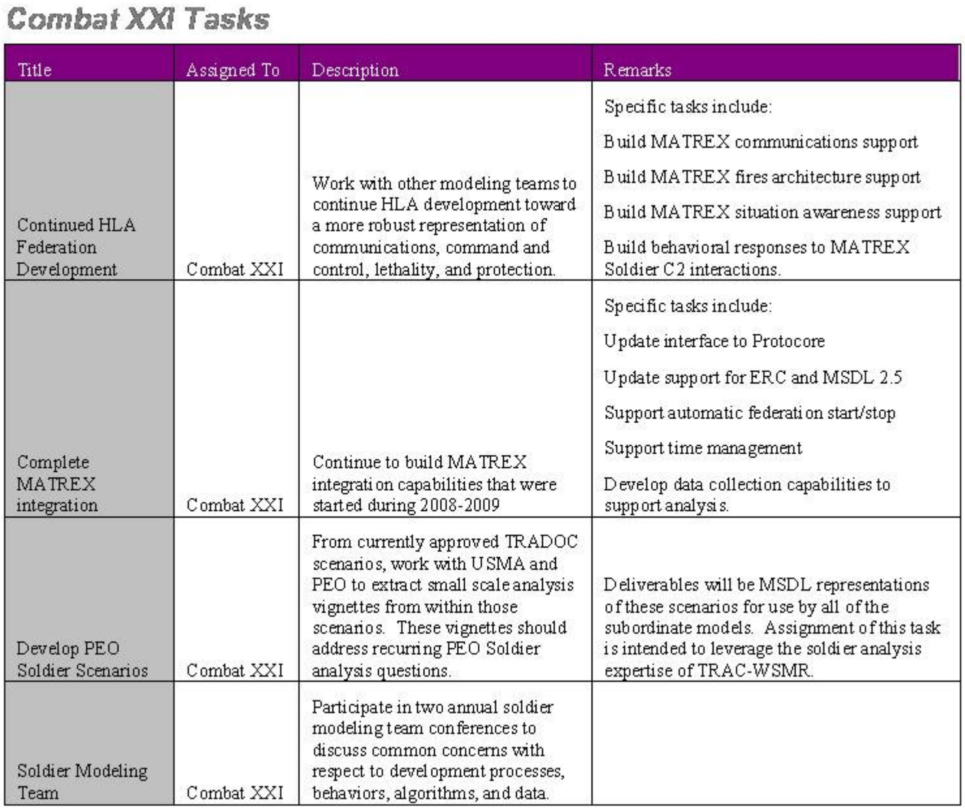
Figure 35: OneSAF tasks for academic year 2009-2010.



America's Army Tasks

Title	Assigned To	Description	Remarks
America's Army Integration	AARDEC	Perform architecture and engineering work for and integration with America's Army that will allow America's Army virtual soldiers to participate in a OneSAF scenario.	Integration effort will include integration of America's Army into the OneSAF SORD, representation of OneSAF damage and dispersion effects in America's Army, and correlation of terrain via correlated data structures and potential use of the OneSAF ERC.
XM-25 Prototype	AMRDEC	Develop an laser-based XM-25 physical weapon prototype that allows a human player to engage targets on a screen with the XM-25. Integrate that weapon into the America's Army software system.	This will allow a human dismounted player to employ the XM-25 in a virtual scenario.

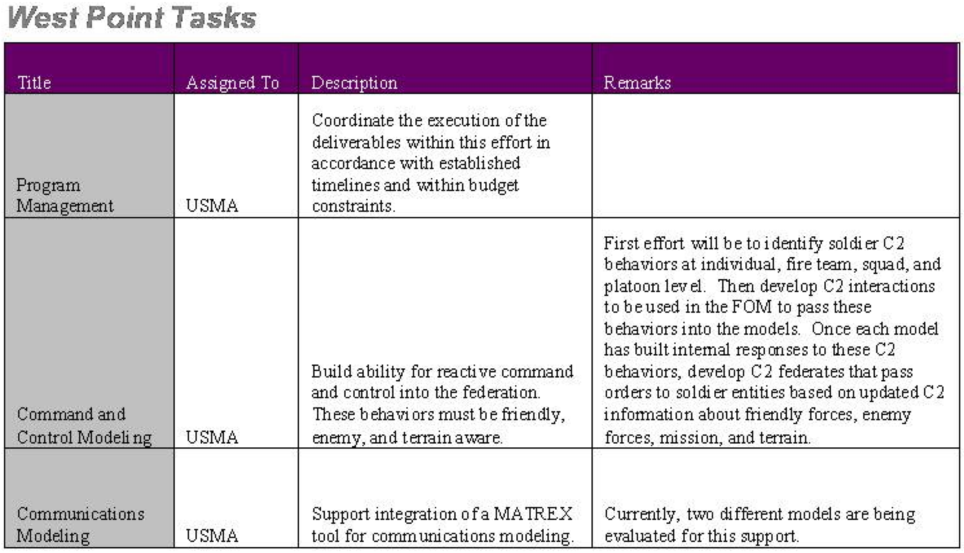
Figure 36: America's Army tasks for academic year 2009-2010.



Combat XXI Tasks

Title	Assigned To	Description	Remarks
Continued HLA Federation Development	Combat XXI	Work with other modeling teams to continue HLA development toward a more robust representation of communications, command and control, lethality, and protection.	Specific tasks include: Build MATREX communications support Build MATREX fires architecture support Build MATREX situation awareness support Build behavioral responses to MATREX Soldier C2 interactions.
Complete MATREX integration	Combat XXI	Continue to build MATREX integration capabilities that were started during 2008-2009	Specific tasks include: Update interface to Protocore Update support for ERC and MSDL 2.5 Support automatic federation start/stop Support time management Develop data collection capabilities to support analysis.
Develop PEO Soldier Scenarios	Combat XXI	From currently approved TRADOC scenarios, work with USMA and PEO to extract small scale analysis vignettes from within those scenarios. These vignettes should address recurring PEO Soldier analysis questions.	Deliverables will be MSDL representations of these scenarios for use by all of the subordinate models. Assignment of this task is intended to leverage the soldier analysis expertise of TRAC-WSMR.
Soldier Modeling Team	Combat XXI	Participate in two annual soldier modeling team conferences to discuss common concerns with respect to development processes, behaviors, algorithms, and data.	

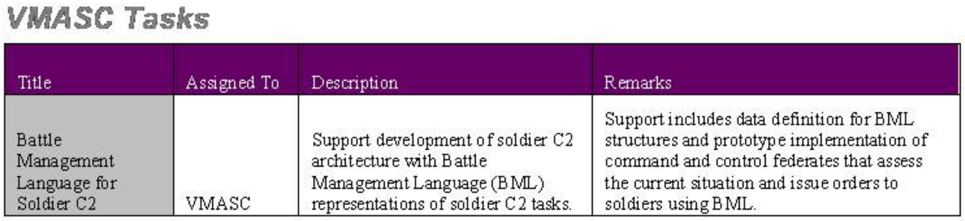
Figure 37: COMBAT^{XXI} tasks for academic year 2009-2010.



West Point Tasks

Title	Assigned To	Description	Remarks
Program Management	USMA	Coordinate the execution of the deliverables within this effort in accordance with established timelines and within budget constraints.	
Command and Control Modeling	USMA	Build ability for reactive command and control into the federation. These behaviors must be friendly, enemy, and terrain aware.	First effort will be to identify soldier C2 behaviors at individual, fire team, squad, and platoon level. Then develop C2 interactions to be used in the FOM to pass these behaviors into the models. Once each model has built internal responses to these C2 behaviors, develop C2 federates that pass orders to soldier entities based on updated C2 information about friendly forces, enemy forces, mission, and terrain.
Communications Modeling	USMA	Support integration of a MATREX tool for communications modeling.	Currently, two different models are being evaluated for this support.

Figure 38: West Point tasks for academic year 2009-2010.



VMASC Tasks

Title	Assigned To	Description	Remarks
Battle Management Language for Soldier C2	VMASC	Support development of soldier C2 architecture with Battle Management Language (BML) representations of soldier C2 tasks.	Support includes data definition for BML structures and prototype implementation of command and control federates that assess the current situation and issue orders to soldiers using BML.

Figure 39: Virginia Modeling Analysis and Simulation Center tasks for academic year 2009-2010.

Distribution List

NAME/AGENCY	ADDRESS	COPIES
Author(s)	Department of Systems Engineering Mahan Hall West Point, NY 10996	2
Client	PEO Soldier 5901 Putnam Road, Bldg 328 Fort Belvoir, VA 22060-5422	5
Dean, USMA	Office of the Dean Building 600 West Point, NY 10996	1
Defense Technical Information Center (DTIC)	ATTN: DTIC-O Defense Technical Information Center 8725 John J. Kingman Rd, Suite 0944 Fort Belvoir, VA 22060-6218	1
Department Head-DSE	Department of Systems Engineering Mahan Hall West Point, NY 10996	1
ORCEN	Department of Systems Engineering Mahan Hall West Point, NY 10996	5
ORCEN Director	Department of Systems Engineering Mahan Hall West Point, NY 10996	1
USMA Library	USMA Library Bldg 757 West Point, NY 10996	1

REPORT DOCUMENTATION PAGE	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE *(DD-MM-YYYY)*
30-10-2009

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
01-06-2008 to 30-06-2009

4. TITLE AND SUBTITLE
PEO Soldier Simulation Road Map VI - Command and Control Modeling

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
LTC Robert H. Kewley, Jr.
Dr. Andreas Tolk

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
USMA Operations Research Center of Excellence, West Point, NY, 10996
Virginia Modeling, Analysis, and Simulation Center, Old Dominion University, Norfolk, VA, 23529

8. PERFORMING ORGANIZATION REPORT NUMBER
DSE-R-0906

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
PEO Soldier
5901 Putnam Road, Bldg 328
Fort Belvoir, VA 22060-5422

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
A - Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT

The purpose of this study is to continue to build a capability for PEO Soldier to assess the platoon level effectiveness of different soldier equipment architectures using distributed simulation. Three combat models - IWARS, OneSAF, and COMBAT XXI - are integrated via the High-Level Architecture (HLA) in order to provide this capability. After last year's effort focused on developing and demonstrating a runtime interaction between IWARS and OneSAF using Research Development and Engineering Command's (RDECOM) Modeling Architecture for Technology, Research, and Experimentation (MATREX) architecture, this year's effort focused on fully developing that architecture for command and control modeling. The resulting federation was able to leverage situation awareness of simulated entities in order to coordinate fires and maneuver in support of a dismounted mission. In addition, a refined systems engineering approach to federation development is proposed using this PEO Soldier federation as an example.

15. SUBJECT TERMS
Distributed Simulation, High Level Architecture, Combat Modeling, Model Driven Architectures, Systems Engineering

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LTC Robert H. Kewley, Jr.
Unclassified	Unclassified	Unclassified	UL	76	19b. TELEPHONE NUMBER *(Include area code)* (845) 938-5529

Reset